Australian Government
Department of Defence
Defence Science and
Technology Organisation

Exploring a Net Centric Architecture using the Net Warrior Airborne Early Warning and Control Node

Kate Foster, Adam Iannos, Geoff Lawrie, Peter Temple and Brad Tobin

Air Operations Division
Defence Science and Technology Organisation

DSTO-TR-2093

ABSTRACT

Network Centric Warfare experimentation is required in order to transform the Australian Defence Force into a net centric force. One area of experimentation is net centric software architectures, particularly component-based systems and middleware. The Airborne Early Warning & Control Mission System Testbed (AEW&C MST) enables such experimentation to be conducted and is overviewed in this report. The AEW&C MST is also one node in the Net Warrior Initiative, which aims to conduct net centric experimentation with real systems, testbeds and simulators across DSTO. This report discusses Net Warrior and the role of the AEW&C MST as the AEW&C node.

Published by

Air Operations Division
DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR-014-085
December 2007

Exploring a Net Centric Architecture using the Net Warrior Airborne Early Warning and Control Node

Executive Summary

Organisations implementing a net centric approach aim to achieve effective and efficient outcomes by capitalising on information sharing for better situational awareness, improved decision making and enhanced collaboration. The main driver for net centricity has been the recent progress achieved in information and communication technologies. These technologies can be considered as essential *enablers* for net centric systems and organisations will be required to adapt their structure and processes in order to exploit them.

Network Centric Warfare (NCW) applies the idea of net centricity to military operations and it is *networking* that underlies the information advantage that NCW may provide. The Australian NCW Concept focuses on an effects-based approach with the aim of increasing operational tempo and improving agility by using information to maximise operational effect and facilitate collaboration.

In alignment with Defence's approach to implementing NCW through 'learning by doing', the DSTO Net Warrior Initiative was conceived to address, through experimentation, new and evolving net centric capabilities and mission system technologies to enhance ADF joint warfighting capabilities. This experimentation will be conducted with real systems, testbeds and simulators across DSTO and, eventually, across Defence. Boeing Australia is also involved in Net Warrior through an Interactive Project Agreement concerning mission systems in NCW environments. Such experimentation will be applied to operational, systems and technical elements of NCW and will enable Net Warrior to provide advice to Defence regarding the extent to which it needs to consider and implement particular NCW concepts and technologies.

One of the nodes in Net Warrior is the Airborne Early Warning & Control Mission System Testbed (AEW&C MST). The AEW&C MST represents the Wedgetail AEW&C capability, which will provide the ADF with an enhanced surveillance and control capability when delivered. The AEW&C MST has been developed to support evaluation of Wedgetail mission computing while providing the freedom to explore the integration of NCW enabling technology into Wedgetail and other platforms. Development of the AEW&C MST and the associated research program is conducted in DSTO's Air Operations Division (AOD) under task 07/044 and is sponsored by DMO. The AEW&C MST is hosted within the AOD Mission System Research Centre (MSRC) along with a range of other mission system testbeds.

The transformation to an Australian net centric force requires a shift in the way systems are procured, built and used, so that information can flow through a changing

network of heterogeneous nodes, each with its own information requirements. For example, aircraft, due to their mobility, will have changing contexts and will require dynamic connections to other nodes. The AEW&C MST provides the infrastructure to conduct research into how information flows can be agile and adaptable in dynamic and distributed environments.

Net centric environments are underpinned by a range of standards and technologies. Such technologies that are important to the Wedgetail capability include component-based systems, Service Oriented Architectures (SOAs), middleware and frameworks. Component-based architectures, supported by middleware and built on top of frameworks, are able to satisfy design needs of applications to produce stable mission and net centric systems. Wedgetail mission computing and the AEW&C MST are built on the Boeing Australia Software Architecture Framework (SAF), a component-based, distributed computing, middleware environment. The SAF employs an SOA approach, with common software component mechanisms and interfaces encapsulated within a patterned framework. The SAF provides access to resources, such as communication through the Common Object Request Broker Architecture (CORBA). SOA concepts, when applied to the needs of net centricity, are able to achieve flexible and adaptable operational effectiveness through the integration of disparate systems and capabilities.

The components of the AEW&C MST are the stimulation environment, mission computing components and monitoring components. The stimulation environment, based on the Engenuity STAGE product, simulates the environment of an AEW&C aircraft and models its sensors. Mission computing components represent adaptations of components from Wedgetail mission computing, while monitoring components provide interfaces for observing the information received and processed by the other components of the AEW&C MST.

Experimentation with the AEW&C MST will investigate technologies that are important for enabling the ADF to become a net centric force. Results from this experimentation will enable Defence to be better informed when acquiring capabilities that interoperate with other systems.

Authors

Kate Foster
Air Operations Division

Dr Kate Foster is a Research Engineer with DSTO's Air Operations Division. Her current work involves support to the Airborne Early Warning & Control acquisition, research into the evaluation of component-based and distributed software architectures, and participation in the DSTO Net Warrior initiative to experimentally investigate aspects of net centricity. Kate obtained a Bachelor of Engineering (Electrical and Electronics) (Hons) and a PhD from Swinburne University in Melbourne, Australia.

Adam Iannos
Air Operations Division

Adam Iannos obtained a Bachelor of Engineering (Computer Systems) (Hons) from the University of Adelaide, Australia in 2001. In 2002, he joined the Air Operations Division at DSTO as a Professional Officer and his work involves support to the Airborne Early Warning & Control acquisition, New Air Combat Capability acquisition, distributed computing design and analysis, and investigating aspects of net centricity.

Geoff Lawrie
Air Operations Division

Geoff Lawrie is the Head of AMS Networks in Airborne Mission Systems Branch within the Air Operations Division of the System Sciences Laboratory, DSTO. He has a degree in Mechanical Engineering. His interests are airborne early warning and control, network centric warfare and advanced information processing.

Peter Temple
Air Operations Division

Peter Temple is a senior research engineer in the DSTO Air Operations Division. He has a Bachelor of Engineering (Electrical) (Hons) and a Bachelor of Science (Mathematics), and has worked for DSTO for 18 years in the field of airborne mission system *integration. His current work includes acquisition and through-life support for the Wedgetail Airborne Early Warning and Control project, and research related to the DSTO Net Warrior NCW initiative.*

Brad Tobin
Air Operations Division

Brad Tobin obtained a Bachelor of Engineering (Computer Systems) (Hons) and a Bachelor of Mathematical and Computer Science majoring in Applied Mathematics and Computer Science from the University of Adelaide, Australia in 2005. In 2006 he joined the Air Operations Division at DSTO as a Research Engineer, where his current work interests are support to the Airborne Early Warning & Control acquisition, the investigation of tactical communication, and aspects of net centricity.

Contents

Acknowledgements

The authors thank Derek Dominish of Boeing Australia for the many discussions regarding the technical content of this report. The effort of Les Vencel of VCORP Consulting, Bill Spencer of DMO, and Lindsay Pears and Brad Wheatley of Boeing Australia in reviewing the report is greatly appreciated.

Abbreviations

ACE	ADAPTIVE Communication Environment
ADAPTIVE	A Dynamically Assembled Protocol Transformation, Integration, and eValuation Environment
ADF	Australian Defence Force
ADGE	Air Defence Ground Environment
ADO	Australian Defence Organisation
AEW&C	Airborne Early Warning & Control
AFRL	Air Force Research Laboratory
AISR ECC	Aerospace Intelligence Surveillance and Reconnaissance Enterprise Capability Centre
AMS	Airborne Mission Segment
AOD	Air Operations Division
API	Application Programming Interface
ATSL	Approved Technology Standards List
AWD	Air Warfare Destroyer
C2	Command and Control
C3ID	Command Control Communications & Intelligence Division
CBSA	Component-Based Software Architecture
CBSE	Component-Based Software Engineering
CCM	CORBA Component Model
CDR	Common Data Representation
CEC	Cooperative Engagement Capability
CFBLNet	Coalition Federated Battle Lab Network
CIAO	Component Integrated ACE ORB
CIOG	Chief Information Officer Group

CLIP	Common Link Integration Processing
CLR	Common Language Runtime
CODEC	Coder-Decoder
COM	Component Object Model
COP	Common Operating Picture
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-The-Shelf
DAF	Defence Architecture Framework
DARPA	Defense Advanced Research Project Agency
DDS	Data Distribution Service
DIE	Defence Information Environment
DII	Defence Information Infrastructure
DIS	Distributed Interactive Simulation
DLS-EC	Dual Link Simulator with Extended Capability
DOC	Distributed Object Computing
DoDAF	Department of Defense Architecture Framework
DSS	Decision Support System
DSTO	Defence Science & Technology Organisation
EO	Electo Optics
ESM	Electronic Support Measures
EWRD	Electronic Warfare & Radar Division
EWSP	Electronic Warfare Self Protection
FCS	Future Combat System
FROG	Forwarding Rules Object Gateway
GIG	Global Information Grid

GIOP	General Inter-ORB Protocol
GMTI	Ground Moving Target Indicator
GUI	Graphical User Interface
IBS	Integrated Broadcast Service
ICD	Interface Control Document
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IDL	Interface Description Language
IFF	Identify Friend Foe
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
IPA	Interactive Project Agreement
ISAR	Inverse Synthetic Aperture Radar
ISRD	Intelligence Surveillance & Reconnaissance Division
JMMTIDS	Joint Moving Map Tactical Information Display System
JORN	Jindalee Operational Radar Network
JSF	Joint Strike Fighter
JTIDS	Joint Tactical Information Distribution System
JTRS	Joint Tactical Radio System
JU	JTIDS Unit
LOD	Land Operations Division
MCDL	Multi-Channel Data Link
MCS	Mission Computing Subsystem
MIL-STD	Military Standard
MOD	Maritime Operations Division

MODAF	Ministry of Defence Architecture Framework
MSI	Multi Sensor Integration
MSRC	Mission System Research Centre
MST	Mission System Testbed
NAF	NATO Architecture Framework
NCW	Network Centric Warfare
NCWPO	Network Centric Warfare Program Office
NCOIC	Network Centric Operations Industry Consortium
NCOW-RM	Network Centric Operations and Warfare Reference Model
NIF	NCOIC Interoperability Framework
NFRM	Netforce Reference Model
NSI	NCW S&T Initiative
OFP	Operational Flight Program
OMG	Object Management Group
OO	Object Oriented
OODA	Observe Orient Decide Act
ORB	Object Request Broker
OS	Operating System
OSI	Open System Interconnection
QoS	Quality of Service
PDU	Protocol Data Unit
RIG	Regional Information Grid
RMI	Remote Method Invocation
RPDE	Rapid Prototyping, Development and Evaluation
RQL	Real-Time Query Language
S&T	Science and Technology

SAD	Situation Awareness Display
SAF	Software Architecture Framework
SAR	Synthetic Aperture Radar
SARM	Strategic Architecture Reference Model
SGT	Scenario Generation Toolset
SIM	Simulation Engine
SIRST	Surveillance InfraRed Search and Track
SM	Scenario Manger
SOA	Service Oriented Architecture
SoS	System of Systems
SoSE	System-of-Systems Engineering
SQL	Sequential Query Language
STAGE	Stimulation Toolkit and Generation Environment
TAAATS	The Australian Advanced Air Traffic System
TAO	The ACE ORB
TCP/IP	Transmission Control Protocol/Internet Protocol
TDF	Tactical Display Framework
TDL	Tactical Data Link
TIE	Tactical Information Exchange
TRM	Technical Reference Model
TTNT	Tactical Targeting Network Technology
UAV	Unmanned Aerial Vehicle
UCAV	Unmanned Combat Aerial Vehicle
UDP	User Datagram Protocol
US DoD	United States Department of Defense

WAN	Wide Area Network
WDL	Weapon Data Link
WIRE	Wedgetail Integration Research Environment
WSD	Weapon Systems Division
WSOA	Weapon Systems Open Architecture
XML	eXtensible Markup Language

1. Introduction

The Australian Defence Force (ADF) is currently implementing the Australian Network Centric Warfare (NCW) concept. In order for this to be successful, the ADF requires advice regarding the underlying technologies that enable NCW. This report is concerned with technologies that facilitate the design and implementation of robust software architectures and mission systems in net centric environments.

Two areas of work are presented that enable experimentation to be conducted with net centric software architectures. The first is the Net Warrior Initiative, which aims to conduct net centric experimentation with real systems, testbeds and simulators across the Defence Science & Technology Organisation (DSTO). Net Warrior is a multi divisional response to the challenge of enhancing the joint warfighting capability of the ADF. Boeing Australia is also involved through an Interactive Project Agreement regarding mission systems in NCW environments.

The second area of work is the Airborne Early Warning & Control Mission System Testbed (AEW&C MST), a component-based and distributed system built on CORBA (Common Object Request Broker Architecture) middleware and the Boeing Australia Software Architecture Framework (SAF). Development of the AEW&C MST and the associated NCW research program is conducted in DSTO’s Air Operations Division (AOD) under the task 07/044 and is sponsored by DMO. The AEW&C MST is hosted by the AOD Mission System Research Centre (MSRC) and allows experimentation into the performance and integration of the mission system for the new Australian AEW&C capability and is a node within Net Warrior.

The motivation for this experimentation is presented in Section 2 with a discussion of NCW and its adoption by Australian Defence. Section 3 discusses Net Warrior, the Wedgetail AEW&C acquisition and the role of the AEW&C MST as the Net Warrior AEW&C node. It is argued that experimentation with high fidelity representations of platforms is required in order for NCW to be successfully implemented.

Technologies and methodologies appropriate to net centric software architectures used in the AEW&C MST include component-based systems, Service Oriented Architectures (SOAs), middleware and frameworks. These are discussed in section 4. Two manifestations of the technologies reviewed in section 4 that are relevant to the AEW&C MST are CORBA and the SAF. Section 5 overviews these and the architecture of the AEW&C MST.

2. Network Centric Warfare and Australian Defence

2.1 Overview

This section introduces the concepts of net centricity and Network Centric Warfare (NCW) with a focus on their application within the Australian Defence Force (ADF) and science and technology for the ADF.

2.2 Net Centricity

A number of definitions for net centricity exist, however many of these have been criticised for lacking clarity [Fewell & Hazen 2003]. Cebrowski [2003] argues that at this stage net centricity is still a concept and, as such, only a working definition can be articulated. Net centricity involves the simple idea that information sharing is potentially of value to an organisation. The aim is to enable the organisation to improve its information position and enhance the capabilities of its decision makers. The net centric approach aims to achieve effective and efficient outcomes by capitalising on information sharing for better situational awareness, improved decision making and enhanced collaboration [Knight *et al.* 2006].

The main driver for net centricity has been the rapid progress achieved in the field of information and communication technologies (ICT) in the second half of the 20th century [Cebrowski & Garstka 1998]. However, merely constructing a superior network is not sufficient for an organisation to become net centric. Net centricity requires an organisation to adapt its structure and processes to exploit ICT and take full advantage of the benefits that it promises. Therefore, communications infrastructure can be considered as an essential *enabler* for net centric systems.

The concept of net centricity has been widely viewed as a network of nodes: a set of nodes (consisting of people, devices, information and services) interact using a communications network to optimise the use of resources and achieve synchronisation of effects. This concept can be applied to various domains, such as commerce, education and military operations.

From a study of the successful usage of the Internet and other effective networks (including social networks), Fewell and Hazen [2003] emphasise the organisational aspects of net centricity and propose the following be included in its definition:

- Nodes may be widely dispersed geographically, but are still able to interact.
- Use of the network is altruistic and each node considers the benefit of its actions to other nodes.
- The network of nodes is considered to be a community and each node, while autonomous, has a sense of responsibility to the community.

The last two points imply that a reasonable amount of trust exists between nodes and Fewell and Hazen [2003, p. 33] argue that '…the difference between a network-enabled application and a net-centric system depends on the relationship between nodes'.

Cebrowski and Garstka [1998] suggest that implementing net centricity requires establishing three grids: an information grid to provide a backplane for computing and communications; a sensor grid to perceive the environment at high speed; and a transaction or engagement grid closely coupled with command and control systems to enable nodes to act efficiently and effectively in the environment.

Keus [2005] uses the simple network-node paradigm as the basis of the Netforce Reference Model (NFRM). In this model, the term *netforce* is used to describe the total collection of nodes that together perform a specific network centric capability, while *network* refers to the communications infrastructure. This is illustrated in Figure 2–1.

The NFRM provides a set of principles with the aim of enabling systems and procedures to be developed in net centric environments. These netforce principles comprise node types, properties and interactions, and netforce functions and services. The NFRM characterises net centricity as both information and network driven:

- Data and information is collected, processed and interpreted.
- Quality information is provided through the network to decision makers.
- Cooperative and synchronised decision making creates tailored measures.
- These tailored measures are executed in a timely, accurate and synchronised manner.

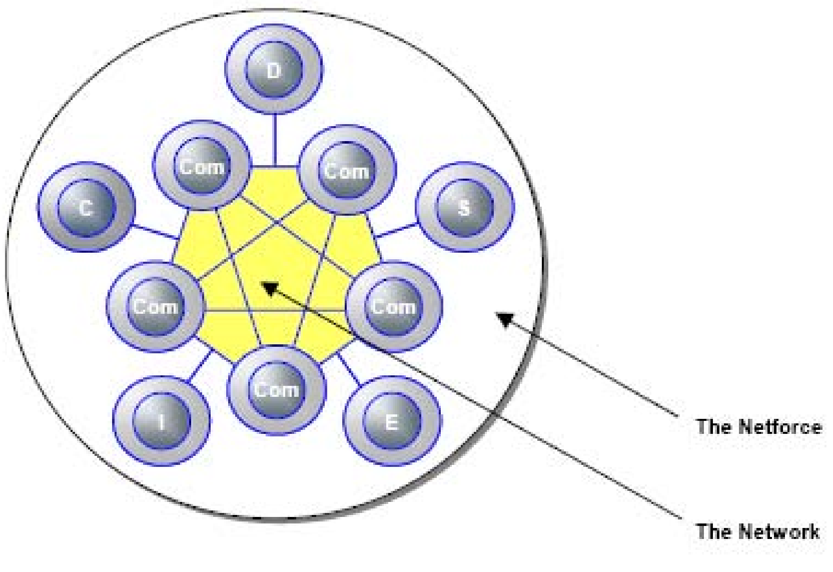

Figure 2–1. The netforce and the network. Source: [Keus 2005]

This characterisation is represented diagrammatically in Figure 2–2, which maps aspects of net centricity to elements of the observe-orient-decide-act (OODA) loop [Boyd 1996]. From this characterisation, the following six basic node types (as shown in Figure 2–1) are defined:

- Collector (C): collects data and information.
- Information Provider (I): manipulates data and information (e.g. processes, interprets, associates, correlates and fuses) and provides that information to other nodes in the required format.
- Decider (D): uses available data and information to decide between different actions.
- Effector (E): puts into effect the decisions that result from the decision making process.
- Communicator (Com): transfers data and information using various means.
- Supporter (S): performs a number of actions that enable net centric operations to be performed.

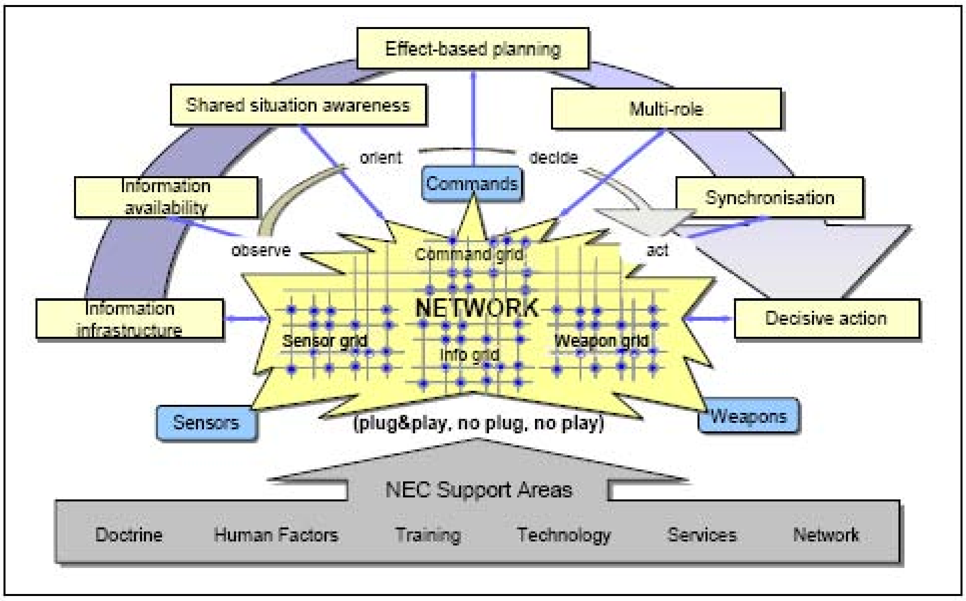

Figure 2–2. Mapping net centricity to the OODA loop. Source: [Keus 2005]

The nature and behaviour of each node is characterised by a number of properties (which may themselves consist of sub-properties). The NFRM specifies the following as a basic set of properties:

- Identity: a unique identifier that distinguishes each node from other nodes in the netforce.
- Status: specifies the operational status of a node.
- Capability: indicates the operational capability of a node. The NFRM takes a Quality of Service (QoS) approach to specifying capability.
- Structure: as nodes may consist of sub-nodes, the structure property details a node's internal structure. For example, an aircraft can be considered as a composite node that consists of sub-nodes representing most of the six basic node types.
- Control: describes the mechanism that is used to control the capability of a node and can be viewed as the node interface.
- Security: specifies what (if any) security aspects relate to a node.
- Integration: indicates how nodes integrate into the netforce.
- Interaction: specifies how nodes interact with other nodes.

Nodes integrate and interface with the netforce through three layers (Figure 2–3). The network communications layer enables a node to connect to the network using security mechanisms if required. The interface layer translates between external and internal node commands and is compatible with a number of approaches to defining generic interfaces for specific objects, e.g. the Object Management Group (OMG). The specification layer takes a QoS approach to specifying node identity and services provided to the netforce. These three layers provide nodes with a standardised interface that has a layered structure. Such an encapsulation mechanism allows legacy systems to be incorporated into the netforce by enabling them to exhibit netforce-compliant behaviour while hiding their internal functions behind the interface.

Simple node interactions can be associated with processes that should occur in net centric environments. For example, interaction between an information provider and a decider is associated with creating situation awareness and interaction between multiple deciders supports synchronised decision making. NFRM functions and services emerge from interactions between the basic node types. For example, Collector Management is a generic NFRM function that employs and controls a number of collectors to optimise activities such as picture compilation, threat evaluation and engagement. Collector Management uses interactions between collectors, information providers, deciders and effectors.

While the NFRM is a generic framework, Keus discusses how its netforce principles can be applied to military operations by linking them to the Network Centric Operations and Warfare Reference Model (NCOW-RM) and the Global Information Grid (GIG). These are discussed in the next section.

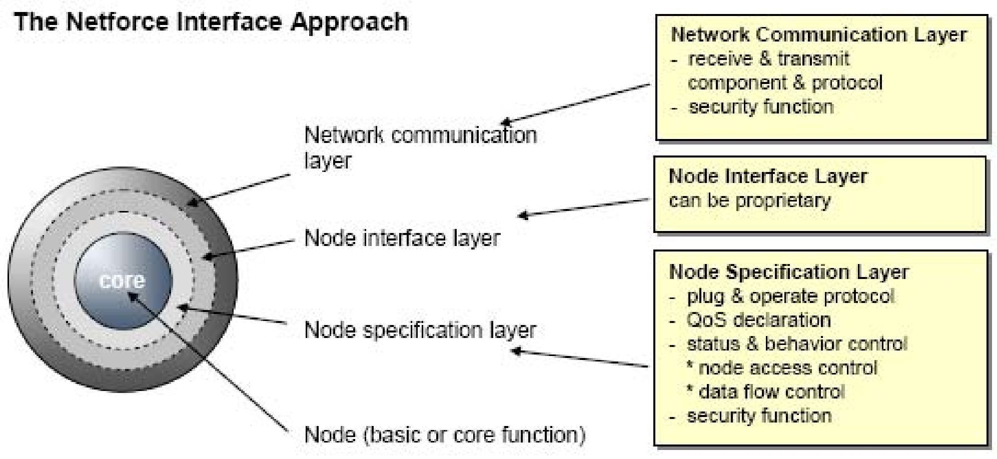
The Netforce Interface Approach
core
Network communication layer
Node interface layer
Node specification layer
Node (basic or core function)
Network Communication Layer
- receive & transmit component & protocol
- security function
Node Interface Layer
can be proprietary
Node Specification Layer
- plug & operate protocol
- QoS declaration
- status & behavior control
* node access control
* data flow control
- security function

Figure 2–3. The layered node structure. Source: [Keus 2005]

2.3 Network Centric Warfare

NCW applies the idea of net centricity to military operations. If a force is able to achieve an information advantage this may translate to a competitive advantage. While the concept of an information advantage is not new, it is *networking* that underlies the information advantage that NCW may provide. NCW, as a distinct concept, first appeared in the public domain in 1998 [Cebrowski & Garstka 1998] as a shift from warfighting based on attrition to a faster and more effective warfighting style. The NCW concept was expanded in the text *Network Centric Warfare* [Alberts *et al.* 1999], which provided the framework for the following tenets of NCW to be developed [OSD 2001, p. 4–1]:

- A robustly networked force improves information sharing.
- Information sharing enhances the quality of information and shared situational awareness.
- Shared situational awareness enables collaboration and self-synchronisation; and enhances sustainability and speed of command.
- These, in turn, dramatically increase mission effectiveness.

NCW involves networking across the physical, information and cognitive domains of military operations [Alberts *et al.* 2001]. The physical domain involves the land, sea, air and space environments and the platforms and communications networks that are situated in these environments. The information domain contains information created by sensing the ground truth in the physical domain and information manipulated by and communicated among warfighters. The cognitive domain exists in the minds of the participants and involves perception, understanding and decision making. The networking of these three domains has the potential to provide secure and seamless connectivity, greater collaboration among the elements of a military force, ubiquitous information, improved situational awareness, synchronisation of operations, increased operational tempo and power to the sharp edge, and improved survivability, lethality and responsiveness.

Realisation of NCW requires technology improvements, the evolution of organisations and doctrine, appropriate tactics, techniques and procedures, and the development of relevant training. One requirement for NCW is an improved capability for operating in the information domain. NCW aims to improve the quality of information and the degree to which information can be shared, thus providing access to the net centric part of the information domain. A concept for achieving this improved capability is the GIG [Alberts & Hayes 2003]. The GIG can be viewed as a mesh of information sources and sinks, through which information is able to be managed and provided on demand to warfighters, decision makers and support personnel. The United States Department of Defense (US DoD) has mandated that the GIG be its technical infrastructure for supporting NCW [US JFC 2001]. The aim is for all relevant information systems, national security systems, advanced weapons platforms, sensor systems and command and control centres to eventually be linked through the GIG.

The development of the GIG is guided by the US DoD enterprise architecture approach, which is represented by the Department of Defense Architecture Framework (DoDAF) [DoDAF WG 2004] and the NCOW-RM [US DoD 2004]. Both DoDAF and the NCOW-RM take a Service Oriented Architecture (SOA) approach to NCW and mandate the use of XML and other web-based standards. While DoDAF and the NCOW-RM specify an *architectural approach*, a number of concrete architectures (e.g. those described by Dekker [2005]) can be derived from the one approach by selecting components and services to meet the requirements of each specific architecture.

Compliance with DoDAF is required and it aims to achieve a consistent architectural model that will enable information sharing and component reuse across the US DoD. The NCOW-RM defines services and standards for the US approach to NCW, which includes business and management operations along with warfighting. The NCOW-RM identifies the following four features of net centric operations: reach, richness, agility and assurance. The intention of this approach is to provide authorised users access to trusted information regardless of time or location.

This section has provided an overview of NCW and its adoption by the US DoD. The next section discusses the Australian approach to NCW.

2.4 Australian Network Centric Warfare

The Australian approach to NCW was officially launched in 2003 by the then Minister for Defence, Robert Hill, in an address to the ADF Network Centric Warfare Conference [Hill 2003]. Australian NCW has been defined as

> …a means of organising the force by using modern information technology to link sensors, decision makers and weapon systems to help people work more effectively together to achieve the commander's intent. [DGCP 2006, p. 5]

Since Minister Hill's address, a number of documents have been produced that provide high level guidance for the implementation of Australian NCW.

The document *ADDP-D3.1 Enabling Future Warfighting: Network Centric Warfare* [DFW 2004] introduces the endorsed Australian NCW Concept. This concept focuses on an effects-based approach for which NCW should contribute at the operational, military-strategic and national levels. The ADF acknowledges that while new concepts and technology will change the character of conflict, the nature of war (e.g. fog, friction and chaos) will endure. Australian NCW aims to increase operational tempo and improve agility by using information to maximise operational effect and facilitating collaboration.

The NCW Concept is a balanced approach in which the human dimension is seen as fundamental to NCW:

> The network is only an enabler to warfighting effectiveness; it supplements but cannot replace the skill, intuition and willpower of the ADF's people. The focus on training, doctrine, leadership and organisation will balance the technical aspects that often dominate discussion of NCW. [DFW 2004, p. 3–1]

The human (or organisational and sociological) dimension is concerned with training, education, doctrine, organisation and leadership and requires trust to enable effective collaboration. The network (or technological) dimension connects engagement, sensor and command systems. A third component, networking, describes how the ADF's human and network dimensions will collaborate to build a system of systems [DGCP 2007].

Therefore, the Australian focus is on the adaptation of military structure, tactics and concept of operations to net centric environments so that greater improvement can be achieved (for a discussion of this applied to the Australian Army see [Krause 2005]). In other words, a key feature of Australian NCW is 'how the user uses the network' [Fewell & Hazen 2003, p. 33].

Five premises have been developed to explain how the human dimension, the network dimension and networking will produce a warfighting advantage. These premises are depicted in Figure 2–4.

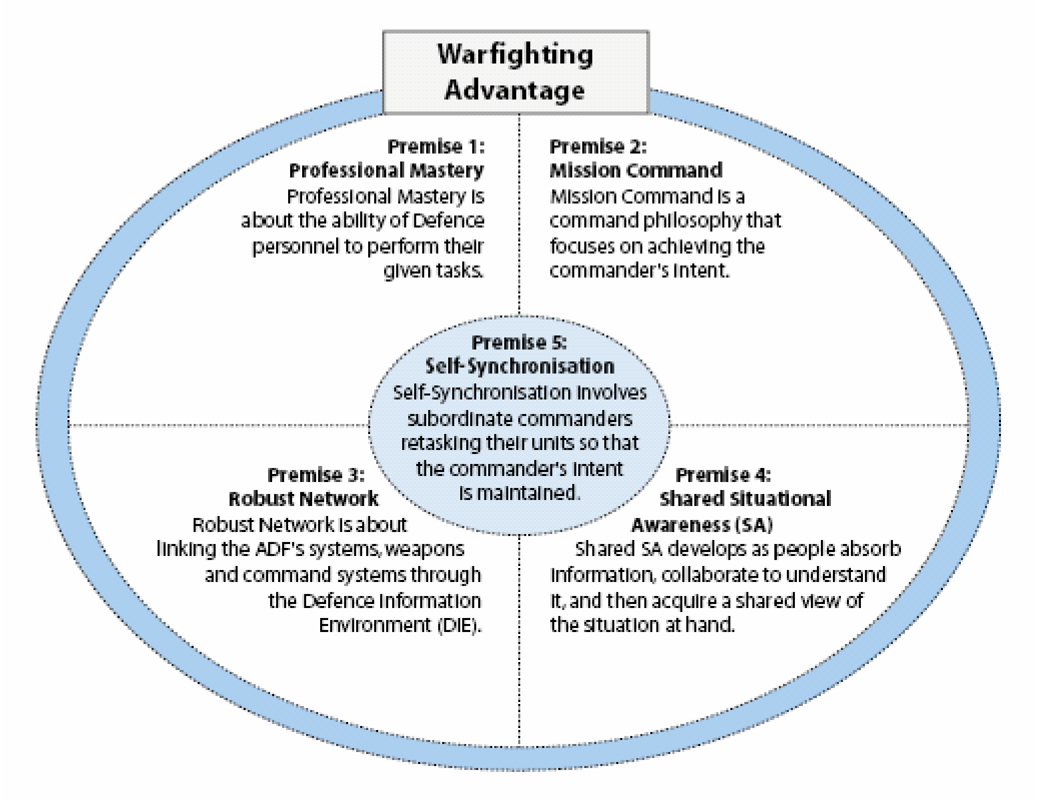
Warfighting Advantage
Premise 1: Professional Mastery
Professional Mastery is about the ability of Defence personnel to perform their given tasks.
Premise 2: Mission Command
Mission Command is a command philosophy that focuses on achieving the commander's intent.
Premise 5: Self-Synchronisation
Self-Synchronisation involves subordinate commanders retasking their units so that the commander's intent is maintained.
Premise 3: Robust Network
Robust Network is about linking the ADF's systems, weapons and command systems through the Defence Information Environment (DIE).
Premise 4: Shared Situational Awareness (SA)
Shared SA develops as people absorb information, collaborate to understand it, and then acquire a shared view of the situation at hand.

Figure 2–4. The five premises of the NCW Concept. Source [DGCP 2006, p. 10]

The following elements have been proposed in order to achieve self-synchronisation (premise 5) and deliver the desired operational effects:

- A sensor grid, which consists of sensors and intelligence sources.
- A C2 grid and an engagement grid will use information from the sensor grid to achieve more effective command, control and targeting.
- An information grid, which is a network that better connects elements of Defence and protects its information.

Each of these grids consists of a human dimension and a network dimension along with a networking component. Figure 2–5 illustrates how these grids will interact. In practice they may not be separate and some systems will consist of a combination of grids.

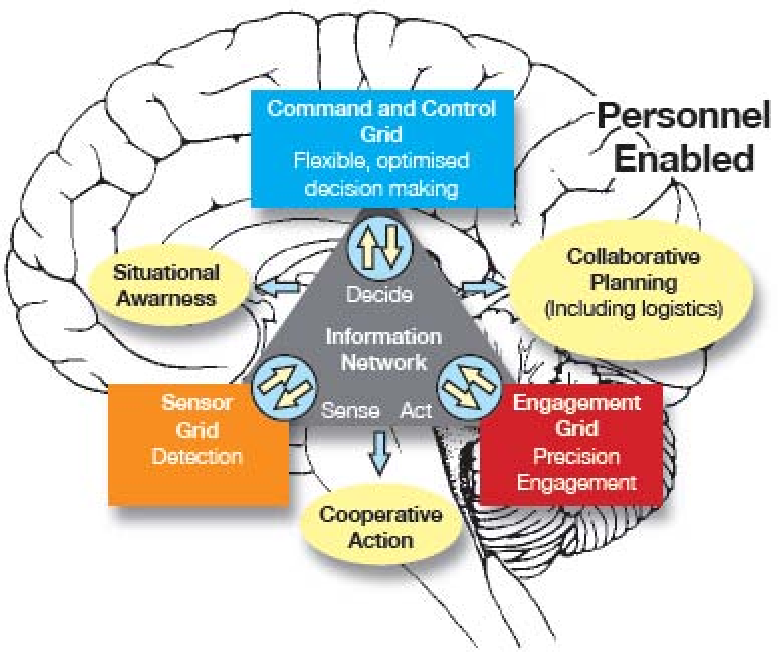

Figure 2–5. Interaction between the key elements of Australian NCW. Source [DGCP 2007, p. 6]

The NCW Concept is the foundation for the *NCW Roadmap* [Director General Capability and Plans 2007], which provides a plan for the implementation of Australian NCW. A 'learn by doing' approach is taken in the NCW Roadmap with the aim of achieving a seamless force in 2020. This is illustrated in Figure 2–6.

Defence has established the Network Centric Warfare Program Office (NCWPO) to monitor and provide support to the development of capabilities for Australian NCW. The NCWPO will achieve this through testing compliance of each capability against constructs such as the Defence Architecture Framework (DAF) and the Approved Technology Standards List (ATSL). These constructs are discussed in Section 4.

While this section has discussed the motivation for and concept of Australian NCW, it is not yet clear how this will be achieved. For example, a challenge is achieving interoperability with our allies. While the aim for the US GIG is to provide a ubiquitous network that enables global connectivity for thousands of nodes, the Australian NCW network will probably have constrained bandwidth and significantly fewer nodes than the US network (a factor of ten or more has been suggested [McKenna *et al.* 2006])[1]. However, key Australian nodes will need to interoperate with US nodes (and those of other allies). Therefore, Defence requires advice regarding the extent to which it needs to consider and implement concepts and technologies that have been adopted by other countries and organisations.

[1] For a discussion of the implications of the GIG to Australian NCW see [Chase *et al.* 2006].

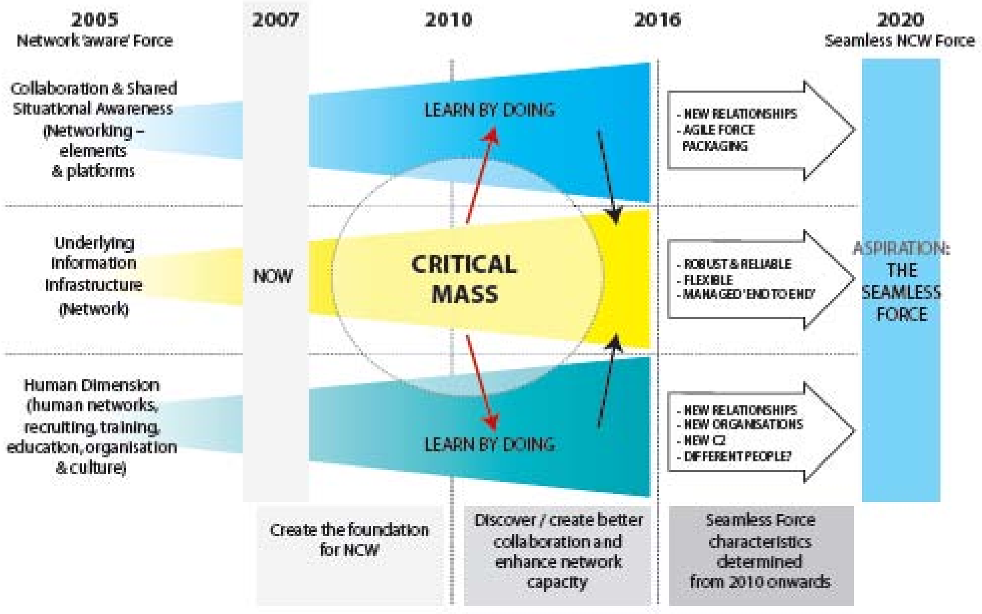
2005
Network 'aware' Force
2007
2010
2016
2020
Seamless NCW Force
Collaboration & Shared Situational Awareness (Networking – elements & platforms
Underlying Information Infrastructure (Network)
Human Dimension (human networks, recruiting, training, education, organisation & culture)
NOW
LEARN BY DOING
CRITICAL MASS
LEARN BY DOING
- NEW RELATIONSHIPS
- AGILE FORCE PACKAGING
- ROBUST & RELIABLE
- FLEXIBLE
- MANAGED 'END TO END'
- NEW RELATIONSHIPS
- NEW ORGANISATIONS
- NEW C2
- DIFFERENT PEOPLE?
ASPIRATION: THE SEAMLESS FORCE
Create the foundation for NCW
Discover / create better collaboration and enhance network capacity
Seamless Force characteristics determined from 2010 onwards

Figure 2–6. The approach taken to develop a seamless force by 2020. Source: [DGCP 2007, p. 20]

In alignment with Defence's approach to implementing NCW by 'learning by doing', one of the aims of articulating the NCW Concept was to provide guidance for Defence's research and experimentation activities [DFW 2004]. Some of these activities are discussed in the next section.

2.5 Science and Technology for Australian Network Centric Warfare

The Defence Science & Technology Organisation (DSTO) is the primary provider of science and technology (S&T) advice to Defence. The DSTO NCW S&T Initiative (NSI) was established in 2004 to coordinate NCW activities across DSTO and foster collaboration for NCW research. The DSTO NSI aims to provide a focal point for NCW research, improve delivery of support to stakeholders, better inform S&T planning and identify areas for further research.

DSTO has produced a significant body of work for Australian NCW. This includes areas such as architectures [Dekker 2005], metrics [Hue 2007], standards [Vencel 2006], compliance processes [Knight *et al.* 2006], modelling and characteristics [Fewell & Hazen 2003], force transformation studies [Chim *et al.* 2007], an Australian Regional Information Grid (RIG) [Chase *et al.* 2006] and force design [DSTO NCW Tiger Team 2 2005]. However, experimentation has been limited. According to Moon, experimentation is

> …of significance and importance not only to the progression of our understanding of the behaviour of complex networks, but also to the application of net-centric approaches to military operations. [Moon 2006, p. 11]

Two recent programs complement the DSTO NSI. The DSTO Experimentation Initiative aims to develop a coordinated approach to experimentation across Defence, and the Defence Rapid Prototyping, Development and Evaluation (RPDE) program works with stakeholders to identify issues and solutions for high priority NCW problems. However, the missing element is a research network of connected battlelabs across DSTO.

DSTO's Net Warrior Initiative was established in late 2005 to connect and conduct net centric experiments with real systems, testbeds and simulators across DSTO and, eventually, wider Defence. Net Warrior will enable the principle of 'learn by doing' to be applied to operational, systems and technical elements of NCW.

2.6 Summary

This section reviewed the origin and concept of net centricity and outlined how this has been applied to develop the NCW approach for the US DoD. The Australian NCW Concept was then discussed along with an overview of the beginnings of its implementation and associated science and technology research and experimentation.

It was argued that DSTO requires a research network of battlelabs in order to support the ADF's move towards a networked force. The Net Warrior Initiative aims to address this requirement and is discussed in the next section along with one of its nodes, which represents the Airborne Early Warning & Control (AEW&C) capability.

3. Net Warrior and the AEW&C Node

3.1 Net Warrior

The Net Warrior initiative in DSTO was conceived to address, through experimentation, new and evolving net centric capabilities and mission system technologies to enhance ADF joint warfighting capabilities. With this as the prime objective, Net Warrior is part of the realisation of a general ambition in DSTO to create a research network of battlelabs. Net Warrior is also a multi-divisional response that supports the DSTO Network Centric Warfare Strategic Initiative (NSI), as discussed in Section 2.5.

The overall purpose of Net Warrior is to contribute to the mitigation of risk to acquisition and implementation of Network Centric Warfare (NCW) and the exploitation of opportunities that NCW presents. Net Warrior will fulfil its purpose if it influences NCW related decisions on defence capabilities and the implementation of NCW in defence.

As a first step, the Net Warrior initiative aims to develop a research capability in NCW by connecting a participating set of nodes that are testbeds representing ADF assets or potential assets in the three domains of air, land and sea. Participating nodes satisfy at least one of the criteria of a) the need for interoperability of the real assets, b) the significance of the real assets in joint operations, c) whether high fidelity representations of the assets exist or are planned in DSTO, and d) whether experimental representations of potential assets would benefit from participating. Seven divisions and Boeing Australia are collaborating in Net Warrior at present. The DSTO divisions which own participating nodes are Air Operations Division (AOD), Maritime Operations Division (MOD), Intelligence Surveillance & Reconnaissance Division (ISRD), Land Operations Division (LOD), Electronic Warfare & Radar Division (EWRD), Weapon Systems Division (WSD), and Command Control Communications & Intelligence Division (C3ID). Other participants from the ADF, industry and academia are likely to join.

Boeing Australia's involvement in Net Warrior is through an Interactive Project Agreement (IPA) under the DSTO/Boeing Australia Strategic Alliance. The IPA, titled *Mission Systems in Network Centric Warfare Environments*, spans the three years until the end of 2009 and defines a collaborative NCW work program. Boeing Australia's interest in Net Warrior is focussed on the analysis of air/ground cooperation and air space management using linked battle management and tactical air operations systems. Linked data systems provide significant opportunity for shared situational awareness; however the operational effects of this type of capability will be seen in the orchestration of air and ground operations, air defence coordination, and air space management.

A characteristic of DSTO nodes participating in Net Warrior at present is that all are high fidelity representations of existing or proposed airborne, land and maritime assets or operational entities. A tenet of the Net Warrior philosophy is that if NCW is to be successfully implemented, NCW concepts and technologies need to be evaluated in environments that closely represent real systems. The nodes exist, in some form, but at present they are not able to interoperate. High fidelity testbeds allow evaluation of real systems and investigation of technical issues. The testbeds will evolve in themselves as integral components of the Net Warrior network and as stand alone components for research capabilities with platform centric research objectives. Where there is common interest, exercises will be run that involve all nodes or a subset. Physical infrastructure is now being rolled out in DSTO that has been specified to satisfy Net Warrior objectives.

A node at Boeing Australia in Brisbane will represent a component from the land domain. The Boeing Australia node will supplement the participating land nodes in LOD. As a result of discussions with Boeing US, a possible future addition is a node at the Boeing US AISR ECC (Aerospace Intelligence Surveillance and Reconnaissance Enterprise Capability Centre) in Seattle, US.

In the Net Warrior context, there are two significant and feasible means of connecting to sites external to DSTO. They are the TDL WAN (Tactical Data Link Wide Area Network), which is now available at DSTO Edinburgh and the CFBLNet (Coalition Federated Battle Lab Network). The two links will provide connectivity between the Net Warrior network and external assets, such as real platforms and battlelabs in other coalition member countries and industry partner facilities.

Through Net Warrior, technological and systems issues can be investigated that are multidisciplinary in nature, such as platform connectivity, mission system integration, multi sensor integration and human system integration. The networked environment will allow emergent properties to be measured and new functions, which may be possible in a networked environment, to be evaluated. Operator in the loop experimentation is envisaged as well as other forms of technical evaluation. The emphasis will be on net centricity and new mission system technologies although it could be regarded as another environment for operations research. Regular coordination meetings aim to identify opportunities for experimentation involving two or more nodes.

3.2 AEW&C Node in Net Warrior

3.2.1 Wedgetail AEW&C

Project AIR 5077, also known as Project Wedgetail, is the acquisition project for Australia's new Airborne Early Warning & Control (AEW&C) capability. The AEW&C capability will provide the ADF with an enhanced surveillance and control capability in the broad expanse of the Australian north. The acquisition contract was signed with The Boeing Company in 2000 with first delivery expected in 2008.

The Wedgetail system consists of the Airborne Mission Segment (AMS), depicted in Figure 3–1, and the Ground Support Segments required for mission support, training and maintenance. Each AMS consists of seven subsystems, as shown in Figure 3–2, which provide the functions of surveillance radar and Identify Friend Foe (IFF), communications, navigation, Electronic Support Measures (ESM), Electronic Warfare Self Protection (EWSP), mission processing, and the Boeing 737 aircraft.

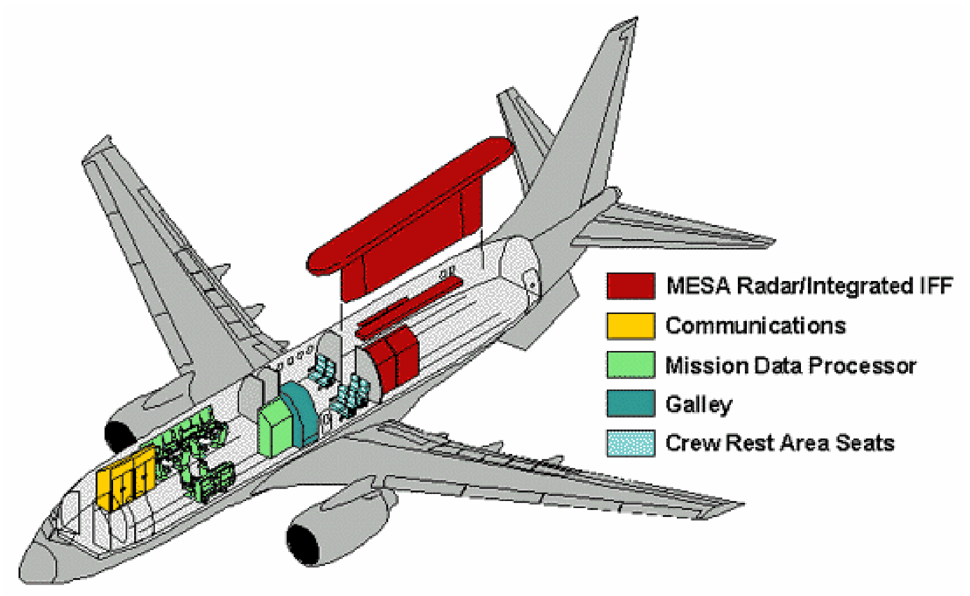

Figure 3–1. The Wedgetail airborne mission segment.

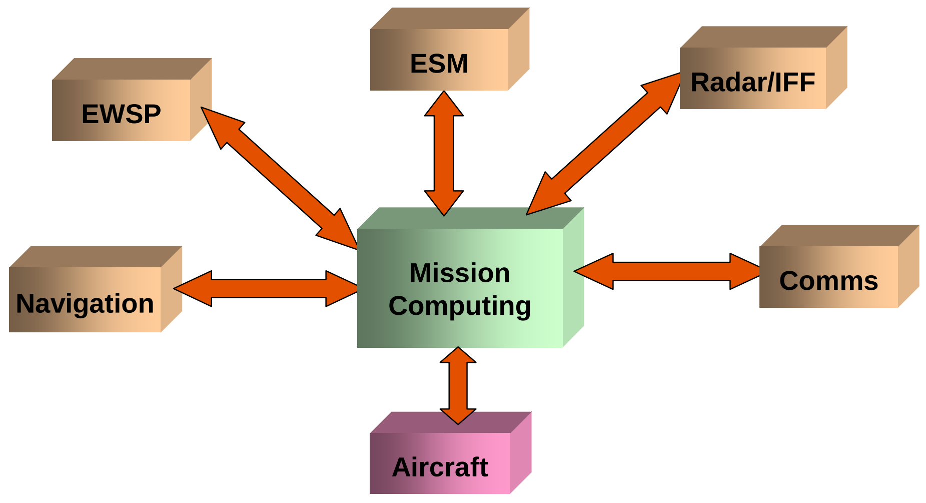

Figure 3–2. Airborne mission segment subsystems.

The Mission Computing Subsystem (MCS) is the critical subsystem at the heart of the Wedgetail mission system. The MCS provides the mission processing for sensor fusion, sensor management, battle management, communications management and system control. The MCS also includes 10 mission consoles and the Flight Deck Tactical Display with associated display processing.

System enhancements using wideband technology will be a vital contributor to allow Wedgetail to be a participant in future NCW but will greatly increase the amount of information that must be processed by the MCS. Software programmable radio technology (such as that provided through the Joint Tactical Radio System (JTRS) program) opens the possibility of innovative approaches to communication requiring extensive software support. New sensors and sensor processing algorithms (for example Ground Moving

Target Indicator (GMTI), Synthetic Aperture Radar (SAR), Inverse Synthetic Aperture Radar (ISAR), Surveillance InfraRed Search and Track (SIRST) and Electo Optics (EO)) will bring new demands on processing. Ongoing improvements to tracking and sensor fusion algorithms (such as Multi-Hypothesis Tracking) and the Human-Machine Interface will further stretch computing resources.

To support these enhanced platform capabilities the MCS will undergo updates throughout the Life of Type of the platform. Traditionally mission systems have been upgraded in major increments, such as mid-life updates. The upgrade philosophy for Wedgetail is ongoing minor increments in capability (Pre-Planned Product Improvement) to allow capability enhancements more in line with operational requirements. This approach of growth in place of wholesale replacement has guided the specification and design for the MCS. Growth options for the Wedgetail capability are further discussed in [Lawrie *et al.* 2005].

3.2.2 AEW&C Mission System Testbed Motivation

As part of the AEW&C integration into the Australian Defence Force, DSTO provides advice in support of the Wedgetail system acquisition, activities leading up to full in-service capability and through life. A number of research areas are relevant to these stages of the AEW&C program, including mission system integration, multi sensor integration, human system integration and platform connectivity and interoperability.

The AEW&C Mission System Testbed (MST) has been developed to support evaluation of the Wedgetail MCS while providing the freedom to develop custom software for NCW experimentation. The AEW&C MST is located within the AOD Mission System Research Centre (MSRC). The MSRC hosts a range of mission system testbeds representing helicopter, fast jet and surveillance aircraft. The focus of the MSRC is on mission system integration, platform connectivity and operator system integration. The MSRC provides an experimentation environment that combines simulation with real system hardware and operator-in-the-loop.

The direction for development of the AEW&C MST and associated research is consistent with a number of drivers. The high level DSTO guidance is to pursue cross divisional coordination and to establish connectivity between DSTO testbeds. Under this vision the testbeds become nodes in a network. Interconnection of DSTO testbeds will provide improved infrastructure to conduct research into cross platform connectivity and NCW operations.

The nature of future defence operations will be based on networking assets and sharing information. Information exchange will be based on more flexible ad-hoc networks. Future tactical networks are likely to make extensive use of the Internet Protocol (IP). In the interim, information exchange using legacy tactical datalinks and tactical datalink message sets will increase.

The ADF's *NCW Roadmap* [DGCP 2007] provides goals for the ADF's NCW capability out to 2020. It identifies capability improvements to the command and control, information, sensor, and engagement grids to enable a more effective networked force.

Alberts *et al.* [1999] identify three forms of experimentation to support the coevolution of NCW: Discovery Experiments; Hypothesis Testing; and Confirming Experiments. Each is essential to the development of capability and methodology supporting NCW, and as shown in Section 2 is lacking from an Australian perspective. Section 2 highlights a vast amount of the work carried out at DSTO in researching concepts, operations and technologies applicable to an NCW environment, however little has been achieved in the way of validating this research. The use of a high fidelity testbed such as the AEW&C MST provides a means for carrying out these three phases of experimentation, as opposed to solely relying on modelling and simulation. Modelling and simulation does however play a role in experimentation by stimulating the AEW&C MST with data to immerse it within a realistic and flexible environment.

3.2.3 AEW&C Mission System Testbed Objectives

The *Future Warfighting Concept* [PGAD 2002] discusses the importance of concept development and experimentation in providing better advice to decision makers. Concept development and experimentation is essential as it reduces risk and enables military innovators to prove and improve their ideas without outlaying significant resources. Concept development gives broad and sometimes ill-defined ideas a chance to be examined by groups of experts in a logical process. However, the results of experimentation must be integrated into the capability development process. *Enabling Future Warfighting: Network Centric Warfare* [DFW 2004] builds on this and discusses the concept of 'learning by doing'. The AEW&C MST is being developed and used to explore NCW concepts through a 'learning by doing approach' where expertise in the concepts, technology, and understanding of emergent properties is gained through experimenting with operational software. In line with this, future research direction will be derived from the outcomes of the initial research.

In broad terms, the AEW&C MST and its supporting research program is under development to support performance evaluation of the Wedgetail MCS and to provide a testbed for exploring the integration of NCW enabling technology into Wedgetail and other platforms.

To assist DSTO in developing research capabilities in these areas, it has been proposed that an AEW&C mission system be acquired to provide a Wedgetail Integration Research Environment (WIRE). This will provide a functionally equivalent subset of the Wedgetail mission system and will be a complementary capability to the AEW&C MST. The WIRE will be used to explore integration issues using functionally equivalent hardware and the actual Operational Flight Programs (OFPs) from the Wedgetail mission system. Of particular importance will be the ability to facilitate operator in the loop experimentation to address Decision Support System (DSS) technology suitability and operator / system interaction optimisation.

It is proposed to link the AEW&C MST and the WIRE with other laboratory environments under the Net Warrior activity. Initial planned demonstrations include 'ping' connectivity, followed by shared Common Operating Picture (COP) using J-series messages and Distributed Interactive Simulation (DIS) to share a scenario. Longer term, regional grid concepts [Chase *et al.* 2006] will be investigated.

AOD research areas which are relevant to AEW&C include: mission system integration; multi sensor integration; human system integration; platform connectivity and interoperability; and architectures and architectural styles appropriate to net centric environments.

Mission system integration research addresses issues associated with the integration of additional systems, with maintenance of mission computing performance baselines and with ensuring that the system is architecturally suited to long term evolution given a rapidly changing technical and operational environment.

Multi Sensor Integration (MSI) research addresses technical issues associated with integration of data from multiple sensors and data sources such as onboard sensors (radar, IFF and ESM), from offboard data sources (data links), from operators and from prior information. MSI functions are tracking, identification, situation assessment, threat assessment, and sensor Management. MSI is implemented with a variety of algorithms to reason in the presence of large amounts of disparate, uncertain data. Consequently, algorithm development is central to MSI research.

Platform connectivity and interoperability research in AOD is addressing issues associated with integration of Tactical Information Exchanges (TIEs) with airborne mission systems. It aims to provide the knowledge to ensure minimal impact of addition of proposed new systems on mission system architectures. System latencies, capacities and quality of service are issues that are inherent to different network and system configurations. Connectivity research aims to address these issues for selected network and system configurations. In particular, it is addressing the performance of gateways which have the potential to minimise integration impact and to solve TIE interoperability and TDL beyond line of sight issues. The potential of future technologies such as Cooperative Engagement Capability (CEC), JTRS, Tactical Targeting Network Technology (TTNT), Weapon Data Link (WDL), Integrated Broadcast Service (IBS), Common Link Integration Processing (CLIP) and Multi-Channel Data Link (MCDL) and their impact on mission systems need to be assessed.

Specific research objectives for the AEW&C MST include:

- Evaluation of net centric connectivity of a Wedgetail platform to:
    - another Wedgetail (e.g. mission cooperation/handover)
    - Air Defence Ground Environment (ADGE)
    - Air Warfare Destroyers (AWD)

- Jindalee Operational Radar Network (JORN)
- Joint Strike Fighters (JSF)
- Orion AP3Cs
- Unmanned Aerial Vehicles (UAV) and Unmanned Combat Aerial Vehicles (UCAV)
- The Australian Advanced Air Traffic System (TAAATS)
- Australian customs/immigration.

- Demonstration of Australian Regional Information Grid (RIG) concepts and architectures.
- Evaluation of emergent properties related to the AEW&C node in the Australian RIG, for example:
    - architectures
    - services (e.g. messaging, collaboration, services management, security, discovery and mediation)
    - protocols (e.g. XML, SOAP, meta-data, trusted filters, GIG services, CODECs and DDS (Data Distribution Service—pub-sub))
    - scalability (how does the system scale with increased information flow?)
    - adaptability/agility (ability to dynamically adapt to changing conditions)
    - information assurance/security
    - information sources and sinks (including analysis of meta-services and meta-data)
    - Quality of Service (QoS) – performance, availability, reliability and modifiability[2].

[2] *Performance* is the ability of a system to allocate its computational resources to requests for service in a manner that will satisfy timing requirements (i.e. latency requirements). Impacts are such things as periodic or aperiodic messaging, synchronous or asynchronous protocols, resource contention and locking, network bandwidth and latency, and asking 'big picture' questions rather than individual requests for data across a network. *Availability* is the long-term proportion of time the system is working and delivering its services. (*Reliability* is the probability a system will not fail over some specified interval of time.) *Modifiability* is the ability of a system to be changed after it is implemented (or deployed).

- Investigation of impact of net centric design and information flows on Wedgetail system performance.

- Investigation of Wedgetail communication upgrade paths and related integration. For example, General Inter-ORB Protocol (GIOP) or IP tunnelling over Link 16, IP gateways over extant radios and JTRS.

The US DoD *Net-Centric Checklist* [DCIO 2004] provides further net centric attributes that may be explored using the AEW&C MST. Experimentation using the AEW&C MST will require an evaluation methodology and metrics appropriate to net centric systems, for example [Hue 2007].

Future defence operations will be based on networking assets and sharing information. Development to this end can be observed in many US programs, perhaps none better demonstrating this fact than the army's Future Combat System (FCS) [US Army 2004]. It consists of a number of manned and unmanned systems, a System of Systems (SoS), connected via a common network to enable improved capability. Data is to be passed through this network in a five layered model, and will make use of standards to conform to the Service Oriented Architecture (SOA) approach of the GIG. Like other defence environments, it will incorporate both existing and future platforms, and thus a number of heterogeneous communication mechanisms such as legacy TDLs will remain initially. The possible techniques for transitioning between these two stages of information sharing thus become crucial, allowing for effective operations to continue with minimal interference.

One transitioning technique has been investigated by the Weapon Systems Open Architecture (WSOA) program [Corman & Gossett 2001] funded jointly by the Air Force Research Laboratory (AFRL), Defense Advanced Research Project Agency (DARPA), and the Open Systems Joint Task Force and will be incorporated into the AEW&C MST. WSOA has introduced the concept of using a TDL as a 'virtual backplane', with a CORBA middleware layer tunnelling mechanism formed on top to achieve greater synergy with a layered communication model (the advantages gained through the use of a middleware layer in this situation are discussed in Section 4). The Common Object Request Broker Architecture (CORBA) Common Data Representation (CDR) provides a machine independent way of representing data, with stubs and skeletons handling requests between objects to simplify application development. Forming a CORBA layer over a TDL is made possible through tunnelling the CORBA pluggable protocols framework. This framework permits the transparent use of custom Object Request Broker (ORB) messaging and transport protocols by CORBA applications. This is particularly important where hard latency and jitter constraints exist, rendering the standard GIOP and Internet Inter-ORB Protocol (IIOP) protocols within an IP communication environment inappropriate.

Permitting information sharing across heterogeneous communication mechanisms is one necessary capability for NCW, however a number of characteristics are required or are desirable for its successful implementation. Adaptability and extendibility are two such characteristics and their support may be investigated through a EUROCONTROL concept, the connector [Ehrmanntraut 2003]. Much like the Defence Information Environment (DIE), the air-ground technologies co-existing in the EUROCONTROL's Air Traffic

Management system are vast and require integration. The connector, an entity that represents the interaction between components in a component-based software architecture (CBSA), encapsulates middleware functionality and separates a component from implementation dependencies. This abstraction hides the complexities introduced through interactions with legacy systems such as TDLs, and thus improves the upgradeability of the system. The connector also permits dynamic component management and linking to improve the adaptability of the system in ever changing network and mission configurations.

The encapsulation of middleware functionality lends itself to the investigation of a multitude of technologies and will therefore prove useful in the Net Warrior Initiative.

3.3 Summary

The transformation to an Australian net centric force will require a shift in the way systems are procured, built and used so that information can flow through a changing network of heterogeneous nodes, each with its own information requirements. For example, aircraft, due to their mobility, will have changing contexts and will require dynamic connections to other nodes. This necessitates that research is conducted into how information flows can be agile and adaptable in dynamic and distributed environments. These environments will be underpinned by a range of standards and technologies (such as component-based architectures and middleware), which are discussed in the next section.

4. Component-based Architectures and Middleware

4.1 Overview

In net centric environments, information interoperability is paramount and the ability to seamlessly share information between and within systems in a timely manner is essential. In order to satisfy these requirements new software design techniques and architectures need to be adopted. Capability procurement had typically concentrated on platforms and usually resulted in stovepiped systems that satisfied a capability gap. In net centric environments capabilities need to be acquired with the ability to interoperate with other systems.

This section aims to describe these new approaches, methodologies and architectures to enable the design of interoperable component-based systems. Section 4.2 describes software engineering approaches relevant to producing distributed interoperable systems, and in particular, component-based software engineering (CBSE) methodologies. It also highlights Australia's Defence Architecture Framework (DAF) as a guide for system designers and a tool for Defence capability managers. Section 4.3 introduces the concepts of architectural approaches and the need for reference models. It describes how Service Oriented Architectures (SOAs) are relevant to net centric systems and outlines the importance of reference models to ensure conformance with a particular architectural approach. Section 4.4 examines the relationship between patterns, middleware and frameworks and how they can be combined to produce architectural environments that support component-based designs and SOAs.

4.2 Engineering Approaches and Methodologies

4.2.1 Evolution of Engineering Approaches

Engineering approaches and methodologies applied to large software systems have evolved over time. Figure 4–1 depicts the timeline of the types of systems developed as software engineering design methodologies have evolved and new approaches have been adopted.

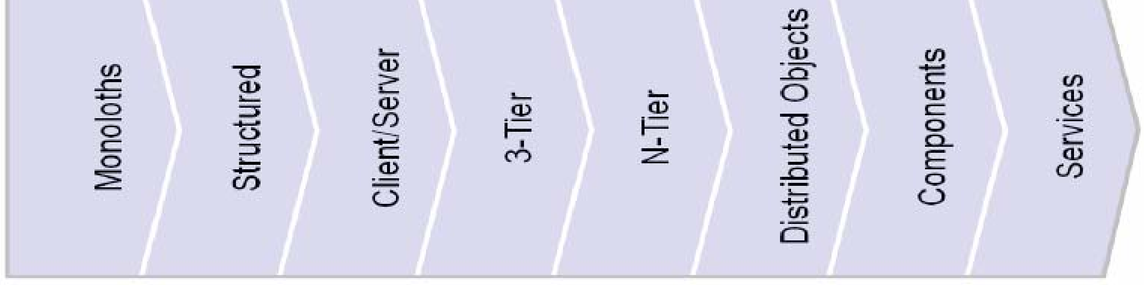

Figure 4–1. Evolution of software architecutres. Adapted from [Cureton 2007 slide 3]

Today large and complex software systems reside on platforms with multiple processors and networks and can span the globe using distributed computing techniques. Many of these systems have been developed in isolation, with little thought of interaction with systems outside of the initial design. Such systems are commonly referred to as *stovepiped systems* and many Defence platforms fit this description. The concept of Network Centric Warfare (NCW) requires improved interoperability between software systems and this can be satisfied by adopting a System-of-Systems Engineering (SoSE) approach.

SoSE is an emerging field of systems engineering with the following goals defined by the System of Systems Engineering Center of Excellence [SoSECE 2007]:

- Individual systems can operate as autonomous components with one or more System-of-Systems (SoS) while satisfying the functional requirements of each system.

- The SoS can explicitly accommodate a wide range of ambiguous and changing conditions.

- The composition of a particular SoS can be reconfigured to form new SoS implementations as conditions demand.

This section explores the DAF, which can be used to document systems and visualise the interrelations of Defence capabilities as a SoS. CBSE is then discussed as it enables systems to be produced that conform with the SoSE approach. CBSE was adopted in the development of the AEW&C node in Net Warrior.

4.2.2 Defence Architecture Framework

The Australian Defence Organisation (ADO) has recognised the need to standardise how Defence describes, models and designs Defence information capabilities within the Defence Information Environment (DIE). The DIE (Figure 4–2) is described as:

> …the aggregate of individuals, their expertise, organisation and systems in the Australian Defence Organisation (ADO) that create, collect, process or disseminate information, including the information itself. [CIOG 2006, Introduction p. 4]

Architecture in relation to the DIE is a disciplined approach to planning, design and implementation of information capability. The Chief Information Officer Group (CIOG) defines this approach to be the DAF and have presented the DAF model, see Figure 4–3. The application of the DAF allows for planners and decision makers to visualise and optimise the DIE as a system of systems. This ensures that through the DIE, the right information is delivered to the right people at the right time to support decision making at all levels.

The DAF has evolved through the combination of elements of the US Department of Defense Architecture Framework (DoDAF) [DoDAF WG 2004] and Metagroup’s (now Gartner) Enterprise Architecture Strategy. While not strictly a SOA approach like DoDAF,

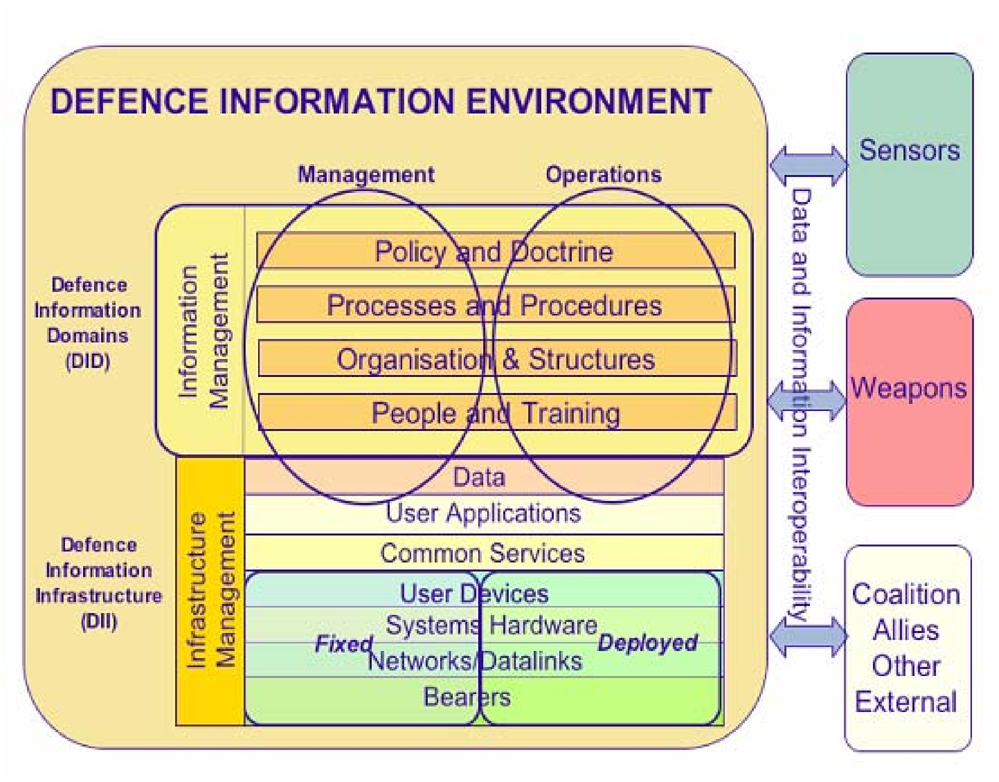

Figure 4–2. Defence Information Environment. Source: [CIOG 2006, Chapter 1 p. 4]

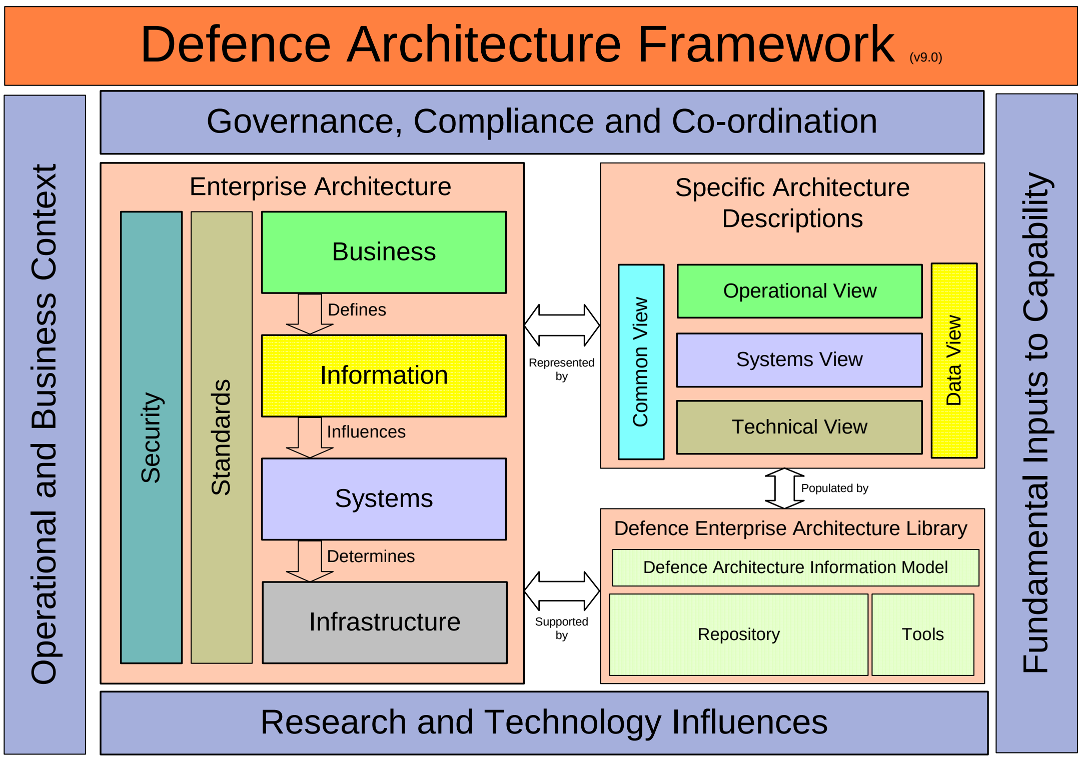
Defence Architecture Framework (v9.0)
Governance, Compliance and Co-ordination
Operational and Business Context
Fundamental Inputs to Capability
Enterprise Architecture
Security
Standards
Business
Defines
Information
Influences
Systems
Determines
Infrastructure
Represented by
Supported by
Specific Architecture Descriptions
Common View
Operational View
Systems View
Technical View
Data View
Populated by
Defence Enterprise Architecture Library
Defence Architecture Information Model
Repository
Tools
Research and Technology Influences

Figure 4–3. Defence Architecture Framework. Source: [CIOG 2006, Chapter 2 p. 1]

the DAF is continually evolving and appears to be more closely aligning with DoDAF. Other information interoperability frameworks in the defence domain include the Ministry of Defence Architecture Framework (MoDAF), the NATO Architecture Framework (NAF) and the Network Centric Operations Industry Consortium (NCOIC) Interoperability Framework (NIF).

Scope for the application of the DAF is broader than technical architecture design as it is applicable from system design through to operations and Defence enterprise business modelling. Thus, the minimum set of outputs defined as essential are those views that are required to define a capability at a high-level to facilitate planning and an understanding of a capabilities place within the DIE.

Experimentation under the Net Warrior Initiative involves the exploration of NCW technologies and techniques, using component-based middleware and frameworks, which is firmly grounded in the Defence Information Infrastructure (DII) component of the DIE.

It is intended that processes and tools from the DAF be used to document the Net Warrior architecture and experiments. As most of the information required to describe the nodes and their interrelations exists in documents produced under Net Warrior, this information could be formally documented using the DAF products to improve common understanding of the architecture in place. The DAF could then be applied to Net Warrior experimental design, with DAF products used to describe the experiments, participants, information and technologies required to generate the required outcome.

4.2.3 Component-based Software Engineering

4.2.3.1 Components

Component-based software engineering is a software design methodology based on the notion of third party composition of software products (components), to produce applications and systems with goals of certifiable, predictable behaviour and quality attributes, with reduced time to market. Benefits of component-based software engineering include:

- software reuse
- possibility of compartmentalised upgrades and maintenance
- allows parallel development
- can improve scalability through the ease of use of replication
- extensibility of a system
- enforced use of standards
- marketplace of components to be assembled.

Components are self-contained and deployable software elements that form applications when assembled with other components. Bachmann *et al.* [2000] propose that components should exhibit the following properties:

- be an opaque implementation of functionality
- be subject to third party composition
- conform to a component model.

The first point implies that components are interchangeable. Any component implementation that satisfies the required behaviour and interface can be substituted for any other. A component can therefore be treated as a 'black box' and users of the component need not rely on the knowledge of the exact implementation details.

The second point represents the need for components to be assembled and deployed into a larger system by any system integrator according to a composition standard. Any system can be comprised of components from a range of sources.

The third point is used to define a component-based architectural design. A conformant component is subject to interface descriptions and architectural constraints imposed by the model. These features enable components to easily interact with other components that conform to the same component model.

4.2.3.2 Component-based Design Pattern

The composition of components to form applications is based on a component-based design pattern [Bachmann *et al.* 2000] realised through the use of well-defined interfaces, conformance to a component model and supported by a component framework. The component-based design pattern (Figure 4–4) comprises software components (1), which are deployable and can be run on a physical or logical device. Components are required to implement one or more interfaces (2) that facilitate conformance to the component model (6). The contractual obligations imposed by the interfaces (3) ensure that independently developed components are able to interact in predicable ways and be deployable in standard build-time and run-time environments (4). Component-based systems comprise specialised component types (5) that perform different roles in the system that are described by interfaces. A component model (6) is the set of component types, their interfaces, and a specification of the patterns of interaction allowed between the components types. A component framework (7) enforces and supports the component model and provides a range of run-time services (8) similar to how operating systems support applications.

4.2.3.3 Component Models

D'Souza and Wills [1999] introduce the concept of a component kit, such that collections of components are designed to work together using a unifying set of principles. This is referred to as a *component architecture type*. Bachmann *et al.* [2000] and Heineman and Councill [2001] further refine this idea to define and report on component models and component frameworks.

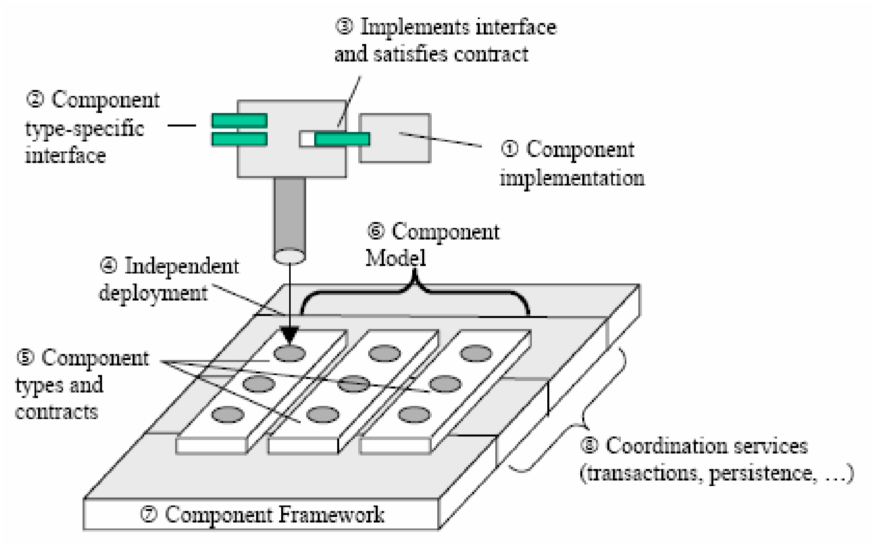
③ Implements interface and satisfies contract
② Component type-specific interface
① Component implementation
⑥ Component Model
④ Independent deployment
⑤ Component types and contracts
⑧ Coordination services (transactions, persistence, …)
⑦ Component Framework

Figure 4–4. Component-based design pattern. Source: [Bachmann et al. 2000, p. 3]

Generally, a component model is the specification of well-defined standards, interfaces and conventions that developers must adhere to when developing components. Conformance with a component model is one property that distinguishes components from other packages of software. Table 4–1 lists the core standards and services required of any component model.

Table 4–1: Basic elements of a component model. Source: [Heineman & Councill 2001, p. 38]

Standards for	**Description**
Interfaces	Specification of component behaviour and properties; Definition of Interface Description Languages (IDL).
Naming	Global unique names for interfaces and components.
Meta-data	Information about components, interfaces and their relationships; API's to services providing such information.
Interoperability	Communication and data exchange among components from different vendors, implemented in different languages.
Customisation	Interfaces for customising components. User-friendly customisation tools will use these interfaces.
Composition	Interfaces and rules for combining components to create larger structures and for substituting and adding components to existing structures.
Evolution Support	Rules and services for replacing components or interfaces by newer versions.
Packaging and Deployment	Packaging implementation and resources needed for installing and configuring a component.

Interfaces

An interface describes a provided behaviour; a user of a component can only rely on the specification of the interfaces that a component supports. The interface acts as a contract between the component and its clients, it describes constraints of a particular service, what the client can expect from the component and what the client needs to provide in turn. A component model may also specify interfaces that a component must implement in order to provide services that the component expects from the run-time environment such as lifecycle management or security. The interfaces that a component implements define the type of the component. If the component implements multiple interfaces, its use can be considered polymorphic and can represent itself as any one of these types.

Naming

Components need to be discoverable and uniquely identifiable. This can be achieved through the use of unique identifiers, naming or directory services. The risk of name clashes can be reduced through the use of hierarchical namespaces.

Meta-Data

Meta-data is used in a component model to provide descriptions of components and interfaces. The model should define how meta-data is described and how to access the data. Examples of meta-data use include Java Beans reflection and introspection and reflection in the Common Object Request Broker Architecture (CORBA) Component Model (CCM) specification.

Interoperability

Interoperability standards define how components communicate with each other and share data. These standards can ensure that components from multiple vendors are able to interact in the same process space, the same machine or over a network. A normalised data representation should be specified to provide a machine independent view of data, which facilitates sharing across a network. For example, the Object Management Group (OMG) has specified the use of Common Data Representation (CDR) in CORBA systems. Standards for Interface Description Languages (IDL) can be used to allow component implementations to be programming language independent. Examples of this include the CORBA IDL specification and Microsoft's CLR (Common Language Runtime) for .NET. Bridging specifications can be defined to allow components designed for different component models to interoperate. For example, the OMG specifies how CORBA components can interoperate with Microsoft COM (Component Object Model) objects and Sun Enterprise Java Beans.

Customisation

Customisation in the context of components is defined in [Heineman & Councill 2001, p. 42] as '…the ability of a consumer to adapt a component prior to its installation or use'. Customisation can be facilitated through the use of specialised interfaces and can be performed using customisation and deployment tools. Customisable aspects of components include properties and behaviour, generally implemented through the use of strategy patterns and policies.

Composition

A fundamental property of component-based systems is the ability to assemble applications from components, potentially sourced from a range of vendors. In order to facilitate this functionality, connector standards are required to enable component connectivity. The two main forms of component connection are asynchronous and synchronous communication methods. Asynchronous interactions are typically based on publish and subscribe mechanisms with event propagation. This method produces loosely coupled systems where the location of event sources and destinations may not be known to either end point. Synchronous communications are based on client/server principles and direct method calls on (potentially distributed) components. This method produces tightly coupled systems that rely on the knowledge of the servant's interface.

Evolution Support

In general, large component-based systems do not remain static. Requirements, interfaces and component implementations can change as new functionality is added. Ideally existing clients of a component whose interface or implementation is modified should be unaffected by the change. It is therefore important that a component model defines rules and standards to enable versioning of interfaces and components.

Packaging and Deployment

Since components are units of standalone deployment, a component model needs to define how a component is packaged as part of the deployment process. Component deployment consists of everything required to install and configure the component within its component framework.

4.2.3.4 Frameworks

A component framework is complementary to the component model and implements infrastructure and services that support or enforce a component model. The framework is similar in concept to an operating system, and provides services and an environment in which components can be deployed and utilised. The framework manages shared resources used by components and facilitates the connections and communications between components. Examples of component frameworks include Enterprise Java Bean servers and containers, and CCM implementations such as the Component Integrated ACE[3] ORB (CIAO). Experimentation in distributed component systems using the Airborne Early Warning & Control Mission System Testbed (AEW&C MST) has explored the use of CIAO and Boeing Australia's Software Architecture Framework (SAF), which is based on open standards.

[3] http://www.cs.wustl.edu/~schmidt/ACE.html.

Component models and frameworks can be general in nature and provide a number of horizontal[4] standards and services. They can also be domain specific and provide a number of vertical standards and services. Horizontal frameworks while more general and applicable to a wider variety of applications normally require more effort from the developer to implement components. Conversely, vertical frameworks simplify the development of components in a particular domain, but are difficult to apply more widely.

In order for components to successfully interact, they need to conform to the same component model or use bridging and adaptation mechanisms to allow interaction across heterogeneous frameworks.

4.3 Architectural Approaches and Reference Models

4.3.1 Software Architecture

There are many definitions of software architecture and many variations on what software architecture entails. The following is a succinct definition:

> Software architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution. [Dikel *et al.* 2001, p. 20]

An architectural approach is more than a design and is broader than an architectural style or specification. The approach guides the architect and developer to design systems according to a high-level concept of organisation and interactions. One architectural approach that is particularly suited to the design of net centric systems is the SOA approach[5]. Reference models assist an architect to design a specific architecture that conforms to an architectural approach.

4.3.2 Service Oriented Architectures

SOAs make software resources available and discoverable as services to end-user applications and other services through public or published interfaces. The most basic goal of SOAs is to implement business processes for enterprise systems. SOAs are able to be applied across enterprise boundaries and are an enabler for the integration of heterogeneous information technology systems.

Net centric operations and warfare require resources to be ubiquitously available within Defence enterprises and across operational boundaries, within security limitations. SOA concepts when applied to the needs of net centricity are able to achieve flexible and

[4] Horizontal descriptions refer to the applicability of the subject to a wide variety of situations and domains, while vertical descriptions refer to subjects that are very domain specific and not applicable outside of the given context.
[5] [Krishnamurthy 2006] provides a detailed comparison of several architectural approaches to designing net centric software systems and their associated frameworks.

adaptable operational effectiveness through the integration of disparate systems and capabilities. For this report the following is adopted as the definition of a service:

> A service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely-coupled (often asynchronous) message-based communication model. [Brown *et al.* 2002, p. 4]

SOAs and CBSE share similar concepts, but services are distinct from components due to their course-grained and discoverable properties. Services generally implement more functionality than components, deal with larger data sets and need to be discoverable at design-time and run-time.

While definitions of SOAs vary, the following key characteristics can be identified [Lewis & Wrage 2004; O'Brien *et al.* 2005; Brown *et al.* 2002; NCOIF 2005]:

- Standards-based interfaces: Services are required to implement at least one interface. The interface acts as a formal contract between the service provider and the service requestor. The use of interfaces standards allows for platform or implementation technology-independent definitions of an interface to facilitate the use of services in heterogeneous environments and is key to achieving the net-centric vision.

- Abstract underlying logic: Services hide their implementation, only the interface and the interface description are made public. Service requestors only rely on the defined behaviour of the interface. This has the advantages of allowing the use of any service that supports the interface, as well as potentially shielding the client from any modifications to the implementation of the service.

- Course-grained: Services usually implement more functionality and operate on larger data sets than components. A service focuses on high level business processes using standard interfaces. If a service is too fine grained, service requestors may need to make more requests than necessary, resulting in inefficient use of resources.

- Loosely coupled: Services are generally connected to other services and applications through standard message-based techniques that reduce dependencies.

- Discoverable: Service interfaces and their descriptions should be discoverable at design-time and run-time and should be understandable to humans and service users. Discovery can be aided through the use of a directory provider or through the use of its network address if known.

- Modular and autonomous: A service represents a boundary around a discrete unit of business logic, and within this boundary, should not be dependent on other services to execute this logic.

- Reusable: Services are designed to support reuse, and use by multiple service requestors.

- Composable: Due to the reusable, modular and course-grained characteristics of services and their implementation of well-defined interfaces, systems and higher-level services can be built through the composition of services and evolved through the addition of new services.

Typical business goals that may lead an organisation to implement a SOA include the ability to be agile and adapt quickly to new opportunities or threats, to reduce costs through streamlining business processes, and removing unnecessary duplication of services. SOAs also introduce the opportunity to share capabilities offered by existing legacy systems.

In order to transition to a SOA, an organisation needs to identify what pieces of functionality or business processes could be represented as services. The granularity of the service needs to be determined, and public or published interfaces need to be designed to expose the functionality. Legacy systems can be incorporated into a SOA through an adaptor. The adaptor is designed to make the legacy system appear as a service by providing a public interface for service requestors to call, while dealing directly with the existing system to access the functionality.

Applications based on a SOA are developed by combining services to realise an emergent behaviour, which is potentially greater than the sum of the parts. Services can be sourced exclusively within the organisation or from external organisations. Each service can also be reused in different applications. This design methodology allows applications to be evolved through the addition of new services.

SOAs require some form of inter-service infrastructure to facilitate interaction and communication between services and applications. Currently, the most common technology used to realise SOAs are Web Services, but this is not the only middleware environment available. J2EE, .NET and CORBA are other commonly used technologies. Importantly, these technologies specify standards for interfaces, communications and data representation, and are all middleware and framework based.

4.3.3 Technical Reference Models

In net centric environments, a Technical Reference Model (TRM) can be described as defining the software components, services and component interactions that may be implemented in a system. To achieve an open systems environment, the layered structure of a TRM aims to ensure separation of data from applications and applications from the computing platform. While a TRM specifies an architectural approach (e.g. a SOA TRM), a number of architectures (e.g. those described by [Dekker 2005]) can be derived from the one TRM by selecting components and services to meet the specific requirements of each architecture. The TRM is analogous to a checklist for conformance of a specific architecture to the architectural approach.

Knight *et al.* [2006] have developed an NCW Enterprise Model that includes a TRM layer and recommend the ADO endorse a TRM with which Defence projects should comply. As one of the aims of Net Warrior is to integrate a set of disparate nodes in a net centric environment, achieving technical interoperability will require adherence to an architectural approach and standards through a TRM. The Net Warrior TRM will provide a mechanism for achieving a common understanding and identifying issues associated with portability, scalability and interoperability.

According to [Vencel 2006], the US and NATO are moving away from their existing platform centric TRMs and are in the process of adopting TRMs that define services and standards for net centric environments. The UK is still developing its architectural approach. The US Network Centric Operations and Warfare Reference Model (NCOW-RM) [US DoD 2004] is more mature than the NATO model, but is still an emerging TRM. Therefore, basing the development of the AEW&C MST (section 5) on, and linking the Net Warrior TRM to, an established industry TRM may be appropriate at this stage.

Boeing in their involvement with the NCOIC has recommended the adoption of the Strategic Architecture Reference Model (SARM) [Logan 2003] or similar model as its TRM. The SARM is a SOA TRM and is consistent with high-level reference models such as the NCOW RM and Open System Interconnection (OSI) model. The SARM is a communication, information, application and presentation architecture framework (Figure 4–5).

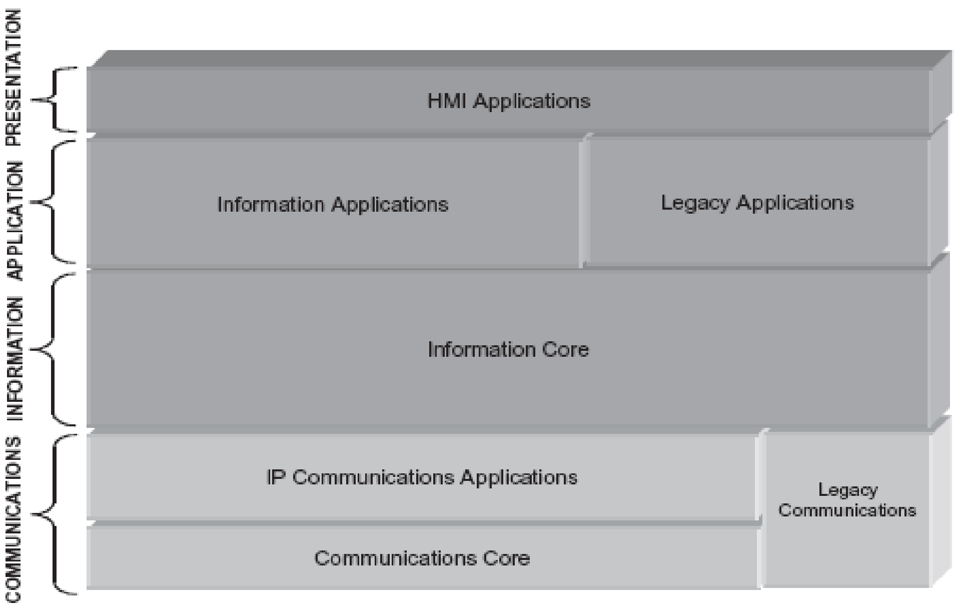

Figure 4–5. Strategic Architecture Reference Model. Source: [Logan 2003, p. 23]

The focus of the SARM is the communication and information layers as these support interoperability between nodes and the application and presentation layers are node specific. The SARM can be decomposed further than Figure 4–5 into a hierarchical collection of components and services based on open standards. Mechanisms for accommodating legacy systems are specified and include adaptors, translators and emulators. The SARM has been adopted in the development of the AEW&C MST and proposed for use as the Net Warrior TRM.

4.4 Reference Architectures, Patterns, Middleware and Frameworks

4.4.1 Reference Architectures

New approaches to designing distributed applications both for net centric and mission system environments are using techniques based on components supported by layered middleware environments that utilise the benefits of frameworks and patterns to produce applications. Boeing's Bold Stroke [Doerr & Sharp 1999; Paunicka *et al.* 2001] and the Weapon Systems Open Architecture (WSOA) program [Corman & Gossett 2001] are examples in the defense domain of experimentation with this technology in the United States.

These approaches conform to CBSE principles by using a layered hardware and software infrastructure that provides a standardised environment in which components can interact with each other and the infrastructure. That is, they describe a component model and provide a framework for conformance. In these environments components are 'plugged into' the underlying infrastructure or *fabric*. The fabric provides a logical connection between components within systems and across system boundaries. It therefore becomes a trade-off of quality attributes and functionality to determine where a component resides rather than tightly coupling the producer and consumer. The use of hierarchical contexts or domains can be employed where logical separation of components or component systems are required.

Figure 4–6 depicts a layered Reference Architecture that supports distributed component systems. The base layer represents the platform environment of operating system and services that run on top of hardware. The communications layer sits above the platform environment and provides standard transport and protocol support. Above the communications layer is the middleware environment that provides operating system abstraction and distribution standards and services to support networked components and systems. Specific domain environments at the top level provide a framework layer to host applications. Components are developed and deployed on top of the domain environment by extending the framework and may interact directly with the middleware environment.

Importantly, standards are relied on at every layer of the reference architecture to provide consistent interfaces between layers and predicable behavior overall. The use of open and established standards at all layers means that a developer mainly needs to design and develop the business logic, while relying on lower layers to provide services for networking, security, lifecycle management and operating system abstraction, among others, while avoiding the need for the developer to write complex, and possibly error prone, 'non-business' logic. Existing systems are not precluded from integration into this layered model. Through the use of adaptors, legacy systems can be presented as a service to other components. The adaptor normalises the legacy system's interface to the fabric and allows the components to effectively and seamlessly 'plug-in'.

The rest of this section will discuss patterns, middleware and frameworks in the context of the Reference Architecture and outline the relationships between these three concepts.

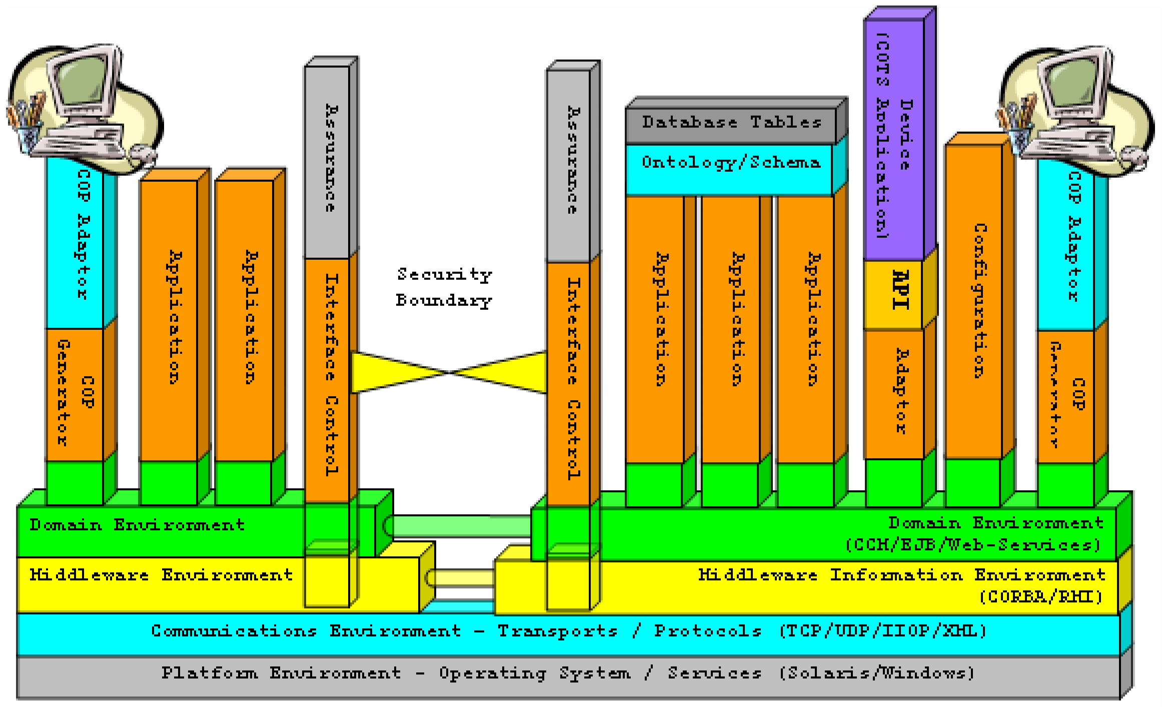

Figure 4–6. Component system reference architecture.

4.4.2 Patterns

Software patterns are solutions to common problems encountered in architecting and designing software. Patterns provide an effective means of communication between software architects, designers and developers. By describing a design using patterns, a common understanding can be imparted of the design problem, its context and an outline of the solution through the description of the structure and dynamics of collaborating classes [Gamma *et al.* 2005]. While not providing strict code reuse, patterns facilitate reuse of the knowledge and experience of previous designs. Patterns themselves are abstract descriptions of problem solutions that need to be implemented by a developer.

There are in general three categories of patterns: architectural patterns, design patterns and idioms. The application of patterns in each category becomes more specialised the further a software system design is delved into.

Architectural patterns [Buschmann *et al.* 1996] solve problems at a system-wide level by describing how elements within a system are organised and structured, and specifying predefined subsystems and their responsibilities. The correct choice of which architectural pattern(s) to implement is essential as it impacts on system-wide attributes.

Design patterns [Gamma *et al.* 1995; Buschmann *et al.* 1996] are applicable for solving design issues at a subsystem level. They describe the components and their relationships to solve a general design problem. Design patterns are smaller in scale than architectural patterns but they are general enough to be paradigm or programming language implementation independent.

Idioms [Buschmann *et al.* 1996] are low-level patterns targeted for a specific programming language. They describe how to implement a component or relationship for a specific language.

No single pattern is able to provide the solution for an entire system. Patterns need to be combined to produce a desired outcome or design and some patterns work together better than others. The relationships between patterns can be represented through a Pattern Language [Schmidt *et al.* 2000]. A pattern language is not a formal language, but a guide to how patterns collaborate. These languages are specific to a particular context. For example, a language of concurrent and networked objects (Figure 4–7) has been used in the development of the AEW&C MST.

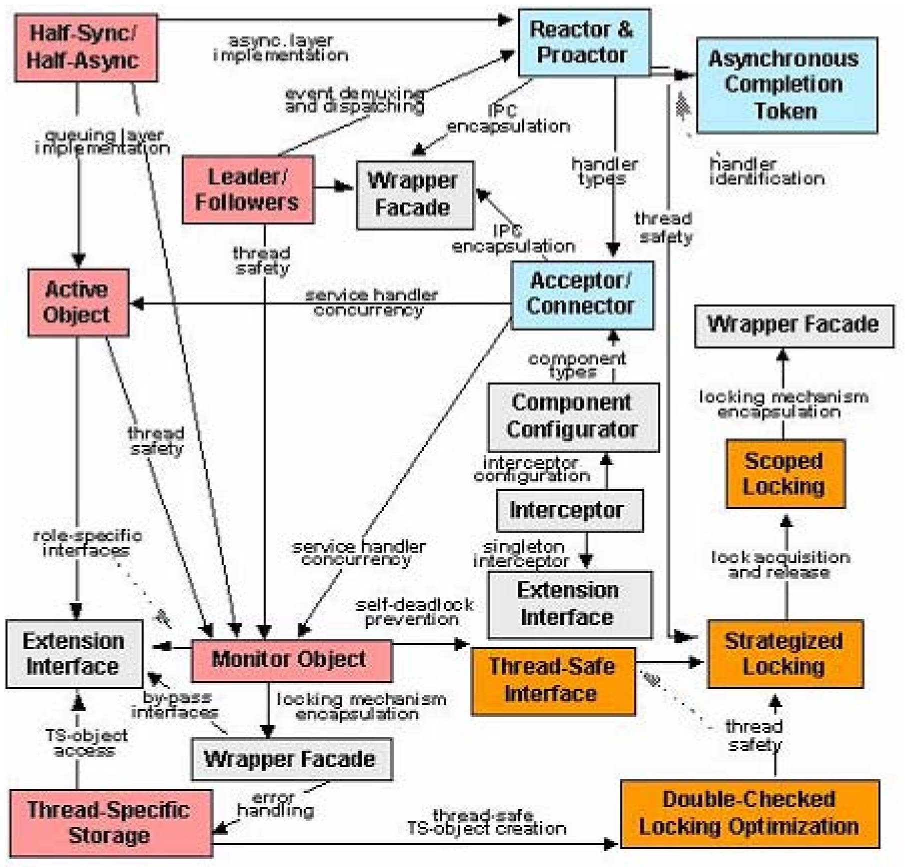

Figure 4–7. Concurrent and networked objects pattern language. Source: [Schmidt et al. 2000, inside rear cover]

4.4.3 Middleware

Middleware is the glue infrastructure that makes distributed component-based systems and SOAs possible. It provides reusable software that functionally bridges two key gaps between (1) end-to-end application functional requirements and (2) the lower-level operating systems, networking protocol stacks, databases and hardware devices [Schmidt & Bushchmann 2003]. The middleware layer can be decomposed into multiple layers (Figure 4–8) much like a network protocol stack, each providing distinct functionality.

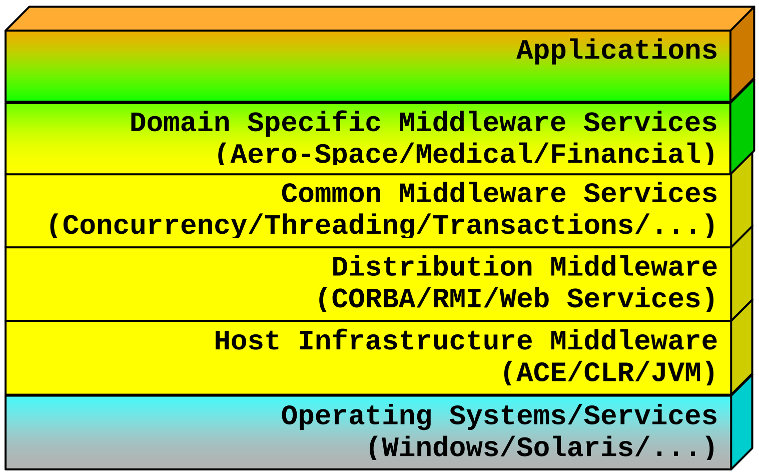
Applications
Domain Specific Middleware Services
(Aero-Space/Medical/Financial)
Common Middleware Services
(Concurrency/Threading/Transactions/...)
Distribution Middleware
(CORBA/RMI/Web Services)
Host Infrastructure Middleware
(ACE/CLR/JVM)
Operating Systems/Services
(Windows/Solaris/...)

Figure 4–8. Middleware layers

Host infrastructure middleware abstracts and enhances operating system (OS) mechanisms to provide reusable interprocess communication, event demultiplexing, concurrency and synchronisation. This layer provides an OS independent environment for using low level OS application programming interfaces (APIs) by encapsulating the peculiarities of an OS and presenting a normalised interface to the layers above.

Common examples of host infrastructure middleware include Sun's Java Virtual Machine, Microsoft's CLR, and ACE. The AEW&C MST uses ACE to provide the host infrastructure middleware in the experimentation environment.

Distribution middleware builds on host infrastructure middleware and defines programming models for distributed computing. Distribution middleware enables clients to invoke methods remotely on a target object in a location independent manner, without depending on hard-coding communication protocols and interconnects, or dealing directly with hardware.

Popular forms of distribution middleware include CORBA, Sun's Java Remote Method Invocation (RMI) and Microsoft's .NET. Common to these technologies, and what enables the distribution, is the reliance on request brokers. Request brokers enable objects to interoperate, usually via proxies, across heterogeneous platforms and networks. Other distribution middleware technologies that are gaining popularity are Web Services, which rely on XML-based SOAP. The AEW&C MST relies on CORBA-based distribution middleware provided by The ACE ORB (TAO)[6].

[6] http://www.cs.wustl.edu/~schmidt/TAO.html.

Common middleware services expand on the capabilities provided by distribution middleware through the definition of domain-independent, reusable software services that are available for application developers. These services negate the need for the application developers to write 'plumbing' code via lower-level middleware that handles distributed resources and allows them to concentrate on writing business logic. Some examples of the types of services typically provided are transactional behavior, location independence, security, fault tolerance, concurrency, scheduling, pooling and threading.

The difference between distribution middleware and common middleware services is that distributed middleware is designed to manage and coordinate end-system resources conforming to a distributed programming model. Common middleware services focus on the allocation, scheduling and coordination of various resources throughout the distributed system in a structured and consistent manner.

Domain-specific middleware services are specific to the requirements of a particular domain, such as telecommunications, commerce, health and aerospace. The previous middleware layers and services mentioned have been general in nature and are applicable and reusable 'horizontally' in many domains. Domain-specific middleware services satisfy 'vertical' markets and product-line architectures. These services are designed to reduce the development effort while increasing the quality of products in a limited field through reuse. Bold Stroke [Doerr & Sharp 1999; Paunicka *et al.* 2001] and WSOA [Corman & Gossett 2001] are examples where domain-specific middleware has been developed to provide component architectures for military avionics mission systems. The AEW&C MST uses Boeing Australia's SAF, which provides domain specific services for real-time, distributed and concurrent components in the mission system domain.

4.4.4 Frameworks

Development of middleware and complex component-based systems would be difficult without the support of frameworks. They facilitate the reuse of design knowledge and code by providing developers with a toolkit of patterns and components geared towards simplifying and standardising programming practices in a particular domain.

Contrary to traditional software design processes where developers create an application from scratch, often resulting in the reengineering of solutions to problems that have already been solved, frameworks provide commonly recurring solutions for infrastructure and services requirements. A framework can be considered to be an incomplete application that can be extended and customised by application developers to create complete applications. That is, a framework is a collection of classes, some complete, that developers can instantiate, and others that are abstract with hook methods that are implemented by developers. Developers have the responsibility of structuring and defining the behaviour of the application, but can call upon the concrete realisations of patterns provided by the framework to achieve application goals. Frameworks differ from middleware in their completeness, role and usage. A framework is extended and customised to produce a complete application, while middleware is a complete application that a developer uses to fulfil the roles described in Section 4.4.3.

The use of frameworks leads to an inversion of control of the software system. Normally when a developer writes an application, they write the body of the code and call on methods or routines from components and libraries. In this case execution is controlled by the developer. However, frameworks provide facilities for event loop execution, demultiplexing and connection of software components. The framework is the body of the application and it invokes methods implemented by the developer to execute the business logic, which results in the framework controlling the flow of execution.

4.5 Summary

Component-based architectures supported by middleware and built on top of frameworks are able to satisfy design needs of applications to produce stable mission and net centric systems. Basing an architectural design on a TRM and in particular the SARM, CBSE concepts have been combined with a layered architectural style to introduce the component-based reference architecture. This model aligns with a SOA approach and particular emphasis is placed on the use of open and well-defined standards at all layers. These systems rely heavily on interface definition and implementation, which facilitates the assembly of applications and systems through the composition of components.

This section has explored aspects of these engineering and architectural approaches and methodologies at a theoretical level and has briefly mentioned experimentation conducted using these technologies and concepts. The next section describes the application through experimentation of these concepts through the implementation of the AEW&C MST.

5. AEW&C Mission System Testbed

5.1 AEW&C Mission System Testbed Architecture

5.1.1 Common Object Request Broker Architecture

The Airborne Early Warning & Control Mission System Testbed (AEW&C MST) is built on a component-based distributed computing framework that incorporates many architectural design patterns. This framework was developed by Boeing Australia and is called the Software Architecture Framework (SAF). The SAF encapsulates the details of the Common Object Request Broker Architecture (CORBA) middleware and transports, and provides a series of services aiding in the production of a reliable and robust distributed computing environment. The SAF is discussed further in section 5.1.2.

The CORBA specification supplies a set of abstractions and services to address the problems associated with distributed heterogeneous computing, which include reliance on programming languages, operating systems, communication protocols and hardware. The CORBA reference architecture (Figure 5–1) provides interface sets linked by an Object Request Broker (ORB) [Schmidt & Buschmann 2003].

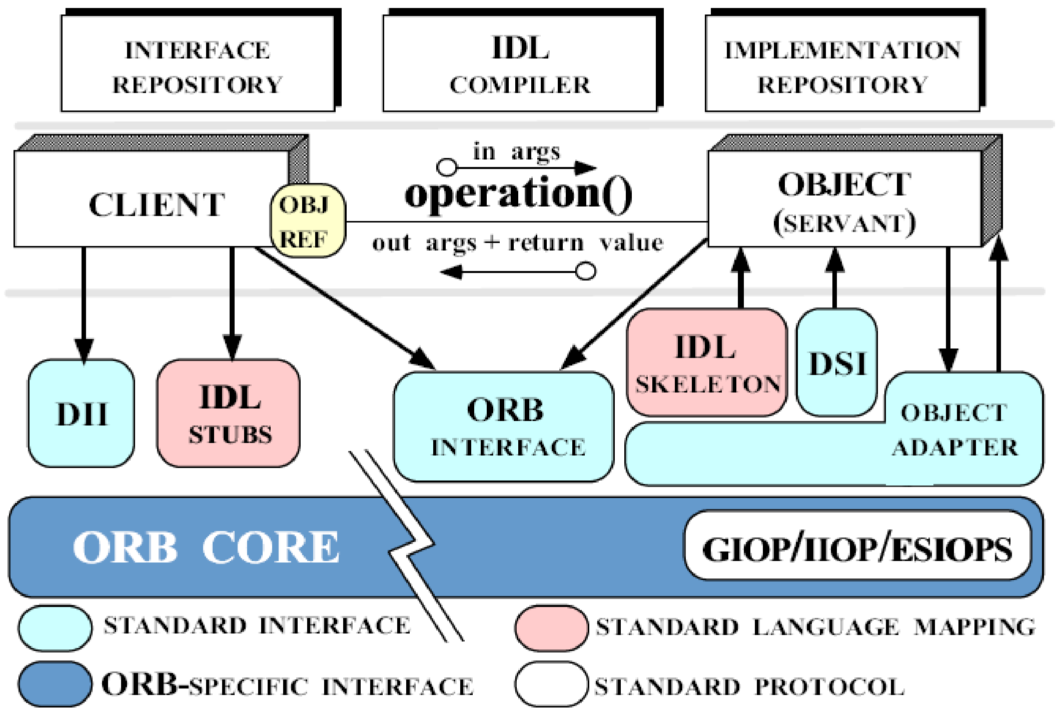

Figure 5–1. CORBA features. Source: [CORBA 2006]

The following description of the features of CORBA is based on [Henning & Vinoski 1999] and [CORBA 2006]:

- Interface Definition Language (IDL): A programming language independent interface that defines supported operations and the data passed to and from these operations.
- Object: An entity that is locatable by an ORB and capable of having client requests invoked upon it.
- Client: A program that makes a request on an object through its defined interface.
- Servant: A programming language entity that implements one or more CORBA objects.
- Stub: A proxy to the servant, generated from IDL, on the client-side. The client makes requests using this stub, which marshals operations into General Inter-ORB Protocol (GIOP) messages in the ORB core.
- ORBs: Enable communication between clients and objects through GIOP messages. Communication takes place in two stages, with the client ORB transmitting requests to a server-side ORB core, which then passes these requests to the object adapter responsible for creating the target object.
- Object adapter: Takes requests dispatched by a server-side ORB core and dispatches them to the skeleton for further delegation.
- Skeleton: A proxy to the servant, generated from IDL, on the server-side. It is responsible for dispatching requests received through the object adapter to the servant implementing the target object.

5.1.2 Software Architecture Framework

The SAF expresses a Service Oriented Architecture (SOA) approach, with common software component mechanisms and interfaces encapsulated into a patterned framework. The general aims of the SAF comply with those outlined in Section 4.4. In SOAs, resources are made available independently through collaborating mechanisms. Requests are made on resources through services, which manage the resources without consideration to underlying platform dependent constraints. This is supported by the SAF through the layered hardware and software infrastructure described in Section 4.4.

The SAF is built on ACE[7], the ADAPTIVE[8] Communication Environment. ACE is an open-source object oriented (OO) framework, developed by the Distributed Object Computing (DOC) Group, that implements many core patterns for concurrent communication software. Like the SAF, the main application of ACE is the development of high

[7] http://www.cs.wustl.edu/~schmidt/ACE.html.

[8] A Dynamically Assembled Protocol Transformation, Integration, and eValuation Environment.

performance, real-time and distributed communication services, with the aim of reducing complexity through higher layer abstractions. Portability is guaranteed by the SAF through ACE's operating system adaptation layer (Figure 5–2), which isolates applications from distinct operating system and network mechanisms by normalising specific operating system differences to standard mechanisms and interfaces. An additional abstraction layer, The ACE ORB (TAO), implements the CORBA middleware specification while utilising the patterns and mechanisms of the lower ACE abstraction.

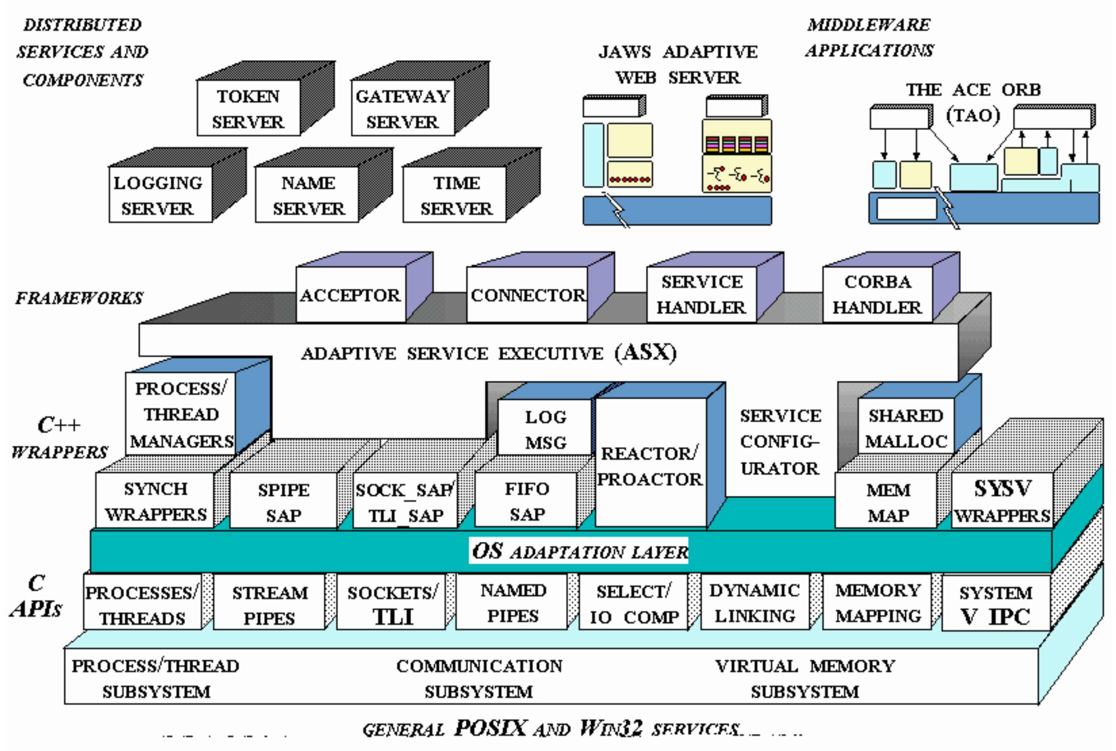

Figure 5–2. ADAPTIVE Communication Environment structure. Source: [ACE 2006]

CORBA is further supported by the SAF through ORB adapter components that ensure compatibility with TAO[9], Orbacus[10] and ORBExpress[11] implementations, making for a pluggable ORB capability. The SAF is currently proven on both Solaris and Windows platforms [Boeing Australia 2005]. The facilities and services provided by the SAF include:

- Core services: The SAF provides two core services, component bus and component model. The component bus encapsulates the CORBA ORB, providing distributed components with context, deployment and configuration services. The component model provides developers with a common interface and interaction model for distributable components.

[9] http://www.cs.wustl.edu/~schmidt/TAO.html.
[10] http://www.orbacus.com/.
[11] http://www.ois.com/products/.

- Streams: Provide a framework for transferring continuous flows of data constrained by Quality of Service (QoS) requirements. The type of data transmitted is defined by IDL and can employ various CORBA transfer mechanisms, including multicast UDP (User Datagram Protocol) and TCP/IP (Transmission Control Protocol/Internet Protocol).

- Events: Support the passing of asynchronous messages between components through event channels. System complexity is reduced by this concept due to the decoupling of event producers from their consumers.

- Domain management: Allows multiple instances of a system to reside on the same computing resources. Each instance can run isolated from the other installations while still providing the capability for resources to be shared.

- Start-up and reliability: Addresses dependency issues between components at start-up, and models the ability of a software component to handle requests at a given time. It also assists with the reliability of components and the ability to handle failures of individual server processes.

- Deployment control: Processes can be configured from multiple sources during deployment and have that configuration managed at run-time. The SAF divides the deployment configuration of processes into properties (attributes of the process and its configuration) and services (the components that are executable within the process space).

- Instrumentation and logging: Log and tracing mechanisms are provided through the command line and configuration files.

A key service provided by the SAF is concurrency. The complexity introduced through concurrency in an OO system is perhaps best summarised by Lea [1999] who defines an OO system as consisting of both objects and activities. These two concepts are interrelated as a '…given object may be involved in multiple activities, and conversely a given activity may span multiple objects' [Lea 1999, p. 38]. Interactions such as these lead to a system with execution that is nondeterministic and thus cannot provide guaranteed *correctness* or *quality*. Four key issues can be identified under these categories:

- correctness
    - safety—nothing bad happens to an object
    - liveliness—something eventually happens within an activity
- quality
    - reusability—the utility of objects and classes across multiple contexts
    - performance—the extent to which activities execute soon and quickly.

A number of constructs exist to support concurrent processing and operation handling. The SAF extensively employs the use of one such construct, known as threads, which provide shared access to resources within a single process. Communication between threads is more efficient than many other concurrency constructs as memory address spaces are not swapped until a process is context switched. However, this introduces its own complexity, requiring further synchronisation and notification patterns to ensure the quality and correctness of code is maintained.

The SAF provides a number of patterns designed to address the four concurrency issues highlighted in [Lea 1999] and thus assists in the development of reliable and robust software. These patterns include latches, barriers, channels, rendezvous, executors, synchronous variables, and a variety of locks, such as mutexes and semaphores. A detailed explanation of these concurrency patterns can be found in [Lea 1999] and [Huston *et al.* 2003].

5.2 AEW&C Mission System Testbed Overview

All custom applications developed for the AEW&C MST take the form of software components[12], primarily based on the SAF component model. This reduces coupling and permits communication via an ORB. At present these custom applications have been developed with the use of CORBA, however any of the technologies described in Section 4 could be used. CORBA has been chosen as it overcomes, through abstraction and services, the problems associated with heterogeneous computer networks [Henning & Vinoski 1999].

The components of the AEW&C MST can be grouped into four main categories (Figure 5–3) and represent: Commercial Off-The-Shelf (COTS); the stimulation environment; mission computing components; and monitoring components. The stimulation environment generates traffic to alter the state of the AEW&C MST. Mission computing represents adaptations of components from the AEW&C Mission Computing Subsystem (MCS), while monitoring components provide interfaces for observing the information received and stored by the other components of the AEW&C MST. The remainder of this section provides a brief description of each component; more detailed descriptions are in Appendix A.

[12] Components are '…units of composition with contractually specified interfaces and explicit context dependencies only' [Szyperski 1998, p. 41].

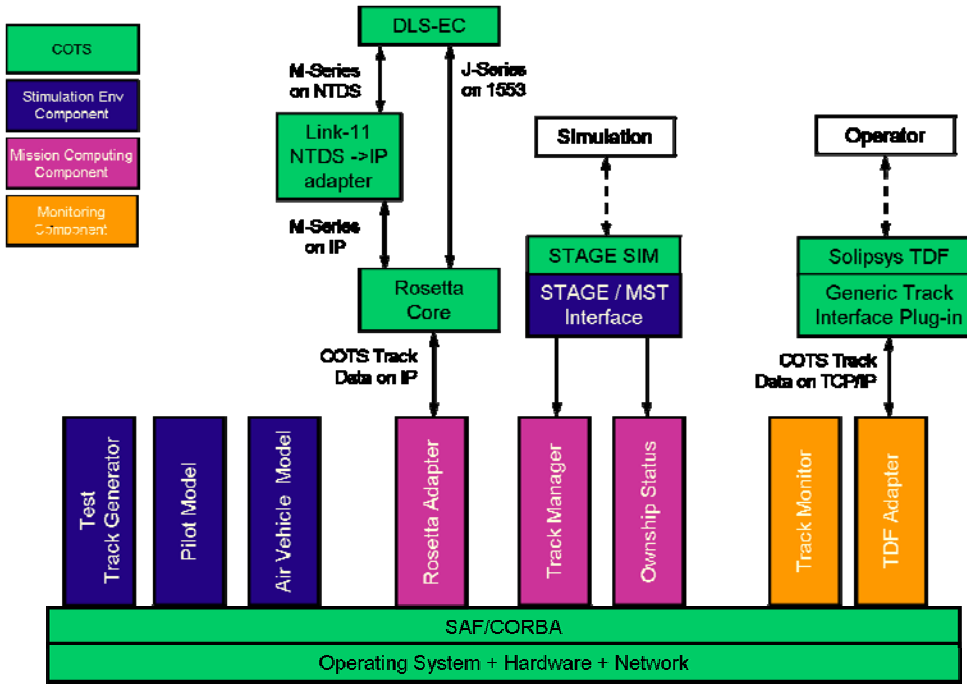

Figure 5–3. The AEW&C Mission System Testbed structure.

The stimulation environment components include the Test Track Generator, Pilot Model, Air Vehicle Model, and the Stimulation Toolkit and Generation Environment (STAGE). All of these components, except STAGE, are simple applications aimed at achieving a specific experimental goal. For example, the Air Vehicle component is a simple model of an aircraft, maintaining aircraft position with basic limits on speed, altitude, acceleration, turn-rate and climb-rate. Such a simple model presents problems for experimentation, but is useful in investigating different designs for data transmission while providing temporary input data with which to test mission computing components. The functionality provided by the stimulation environment components has been improved by the inclusion of the COTS product STAGE. Through the use of the STAGE/MST Interface, which takes data from STAGE's internal data structures and outputs the information in a format compatible with the AEW&C MST, the AEW&C MST is able to use STAGE for all ownship sensor and flight modelling. Currently, this provides kinematic data from an ownship flight model, while an ownship radar model provides target data for all Distributed Interactive Simulation (DIS) entities it detects.

Of particular use in an experimentation context are the mission computing components, which are the main utilities for storing and processing local and remote information. Within the AEW&C MST the Track Manager is responsible for maintaining the tactical state of the AEW&C aircraft. It does so through a common repository of tracks and a collection of capabilities that can be applied to these tracks. The Ownship component complements the Track Manager by maintaining the kinematic state of the AEW&C aircraft.

In addition to maintaining the state of the AEW&C MST, the mission computing components are responsible for interfacing with the tactical data link gateway, Rosetta. To achieve this goal two separate components have been created. One transmits the AEW&C MST's ownship kinematic state while the other is responsible for receiving tactical data at periodic intervals.

The interactions of the mission computing components as well as those of other components are depicted in Figure 5–4. The existing components of the AEW&C MST, and their interactions with internal and external systems are shown in black, while those components and interactions in blue represent items yet to be developed. This distinction highlights the development still required on the tactical data link interface. Tracks contained within the Track Manager are yet to be forwarded to Rosetta, limiting the tactical information communicated to other systems such as the Dual Link Simulator with Extended Capability (DLS-EC). Also separated in Figure 5-4 are the components contributing to the AEW&C MST's involvement in modelling and simulation and the components directly related to mission computing.

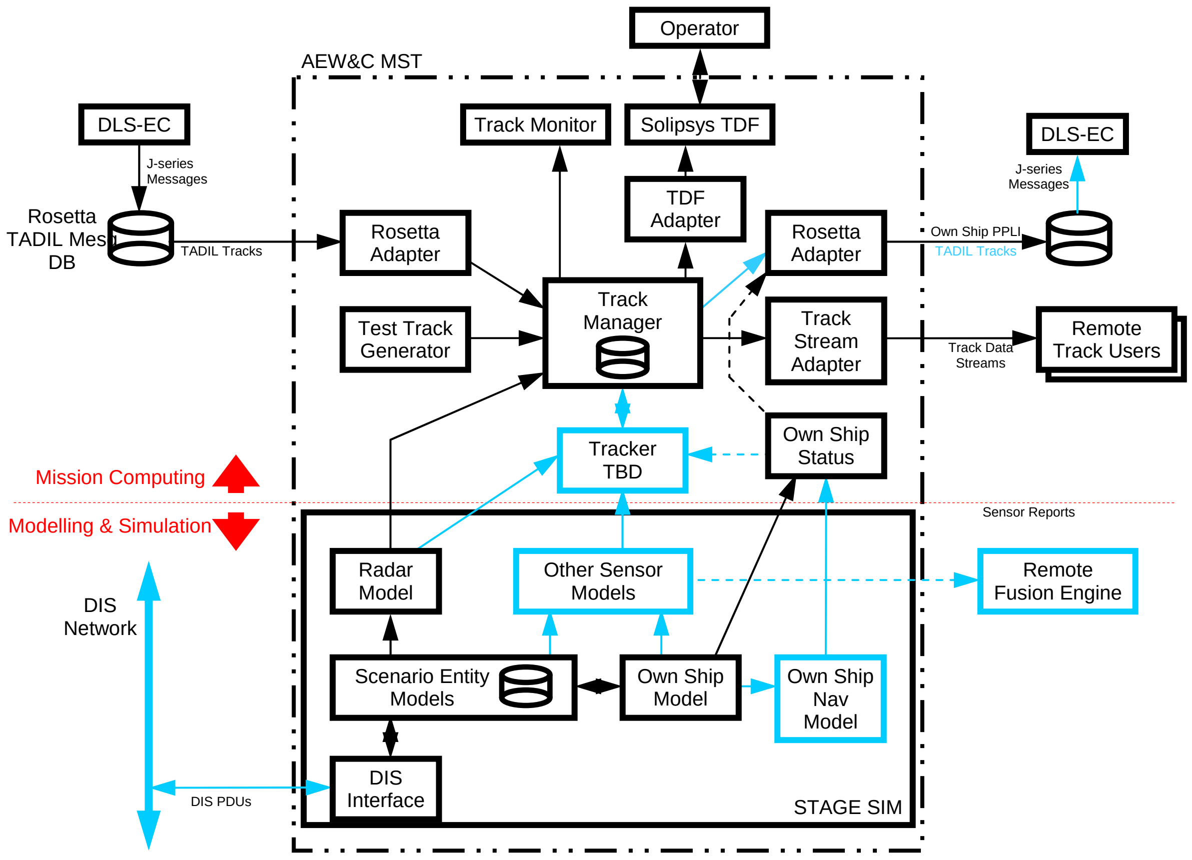

Figure 5–4. The AEW&C Mission System Testbed component interactions.

6. Summary

The transformation of the Australian Defence Force (ADF) into a net centric force requires the adaptation of the way systems are procured, built and used. This report has argued that in order for this transformation to be successful, experimentation is required with the technologies that underpin Network Centric Warfare (NCW).

The Net Warrior Initiative is enabling such experimentation to occur by implementing a network of real systems, high fidelity testbeds and simulators across DSTO and wider Defence. Net Warrior aims to address new and evolving net centric capabilities and mission system technologies to enhance the joint warfighting capability of the ADF. Multiple DSTO divisions are participating in Net Warrior and Boeing Australia is involved through an Interactive Project Agreement concerning mission systems in NCW environments.

The Airborne Early Warning & Control Mission System Testbed (AEW&C MST) is one of the nodes in Net Warrior and has been developed to support the Wedgetail AEW&C capability. The AEW&C MST supports the evaluation of Wedgetail mission computing and enables the exploration of technologies that are relevant to net centric software architectures and mission systems.

Technologies that enable robust mission systems to be developed and implemented for NCW environments include component-based systems, Service Oriented Architectures (SOAs), middleware and frameworks. The AEW&C MST is built on such technologies through the use of the Common Object Request Broker Architecture (CORBA) and the Boeing Australia Software Architecture Framework (SAF). Experimentation using the AEW&C MST will provide insight into how information can be agile and adaptable in NCW environments. Such experimentation will enable DSTO to provide advice to Defence regarding the implementation of particular NCW concepts and technologies, and the acquisition of systems and platforms that interoperate.

7. References

ACE 2006—*see* Overview of ACE (2006).

Alberts, D.C., Garstka, J. J. & Stein, F.P. (1999) *Network Centric Warfare: Developing and Leveraging Information Superiority*, 2nd edition, DoD C4ISR Cooperative Research Program (CCRP), Washington, DC, United States.

Alberts, D.C., Garstka, J. J., Hayes, R. E. & Signori, D. A. (2001) *Understanding Information Age Warfare*, DoD C4ISR Cooperative Research Program (CCRP), Washington, DC, United States.

Alberts, D.C. & Hayes, R. E. (2003) *Power to the Edge*, DoD C4ISR Cooperative Research Program (CCRP), Washington, DC, United States.

Bachmann, F., Bass, L., Buhman, C., Comella-Dorda, S., Long, F., Robert, J. Seacord, R. & Wallnau, K. (2000) *Volume II: Technical Concepts of Component-Based Software Engineering*, 2nd edition, CMU/SEI-2000-TR-008, ESC-TR-2000-007, Carnegie Mellon Software Engineering Institute, Pittsburgh.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (1996) *Pattern-Oriented Software Architecture: A System of Patterns*, vol. 1, Wiley.

Boeing Australia (2005) *Software Architecture Framework*, Brisbane, Australia.

Boyd, J. R. (1996) *The Essence of Winning & Losing* (accessed 2 April 2007), URL: http://www.belisarius.com/modern_business_strategy/boyd/essence/eowl_frameset.htm.

Brown, A., Johnston, S. & Kelly, K. (2002) *Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications*, Rational Software Corporation.

Cebrowski, A. K. & Garstka, J. J. (1998) Network-Centric Warfare: Its Origin and Future, *U. S. Naval Institute Proceedings Magazine*, January.

Cebrowski, A. K. (2003) Network-Centric Warfare: An Emerging Response to the Information Age, *Military Technology*, MILTECH 5/2003, pp. 16–22.

Chase, G., Dall, I. & Gani, R. (2006) *Implications of the Global Information Grid for Australian Network Centric Warfare*, DSTO-TN-0697, Defence Science & Technology Organisation, Edinburgh, Australia.

Chief Information Officer Group (2006) *DIEMAN*, vol. 3, part 1, Department of Defence, Canberra, Australia.

Chim, L., Moon, T., O'Brien M. & Robinson, K. (2007) *Networked Joint Task Force (NJTF) 2015 Study: Definition and Concept*, DSTO-TR-1952, Defence Systems Analysis Division, Defence Science & Technology Organisation, Edinburgh, Australia.

CIOG 2006—*see* Chief Information Officer Group (2006).

CORBA 2006—*see* Overview of CORBA (2006).

Corman, D. & Gossett, J. (2001) Weapons System Open Architecture – Using Emerging Open System Architecture Standards to Enable Innovative Techniques For Time Critical Target Prosecution, in the *Proceedings of the 20th Digital Avionics Systems Conference*, 14 to 18 October, pp. 9.E.4–1 to 9.E.4–8, IEEE.

Cureton, K. (2007) *Overcoming Roadblocks to Interoperability Using NCOIC Tools: Service Oriented Architectures*, Network Centric Operations Industry Consortium, United States.

DCIO 2004—*see* Defense Chief Information Officer (2004).

Defense Chief Information Officer (2004) *Net-Centric Checklist Version 2.1.3*, Department of Defense, Washington, DC, United States.

Dekker, A. (2005) A Taxonomy of Network Centric Warfare Architectures, *Proceedings of the Systems Engineering, Test & Evaluation Conference (SETE)*, Brisbane, Australia, 7 to 9 November.

DeMichiel, L. & Keith, M. (2006) *Enterprise JavaBeans Version 3.0 (Specification)*, Sun Microsystems.

DFW (2004)—*see* Directorate of Future Warfighting (2004).

DGCP (2006)—*see* Director General Capability and Plans (2006).

DGCP (2007)—*see* Director General Capability and Plans (2007).

Dikel, D., Kane, D. & Wilson, J. (2001) *Software Architecture Organizational Principles and Patterns*, Pretence Hall, 2001.

Director General Capability and Plans (2006) *Explaining NCW*, Defence Publishing Service, Department of Defence, Canberra, Australia.

Director General Capability and Plans (2007) *NCW Roadmap 2007*, Defence Publishing Service, Department of Defence, Canberra, Australia.

Directorate of Future Warfighting (2004) *Enabling Future Warfighting: Network Centric Warfare*, ADDP–D.3.1, Defence Publishing Service, Department of Defence, Canberra, Australia.

DoD Architecture Framework Working Group (2004) *DoD Architecture Framework Version 1.0 Deskbook*, Department of Defense, Washington, DC, United States.

DoDAF WG (2004)—*see* DoD Architecture Framework Working Group (2004).

Doerr, B. & Sharp, D. (1999) Freeing product line architectures from execution dependencies (avionics software), in the *Proceedings of the 18th Digital Avionics Systems Conference*, 24 to 29 October, pp. 9.C.2–1 to 9.C.2–8, IEEE.

D'Souza, D. & Wills, A. (1999) *Objects, Components, and Frameworks with UML: the Catalysis Approach*, Addison-Wesley Longman Publishing.

DSTO NCW Tiger Team 2 (2005) *Designing a Force for Network-Centric Warfare*, DSTO-GD-0431, Defence Science & Technology Organisation, Edinburgh, Australia.

Ehrmanntraut, R. (2003) System-of-System Integration of Air-Ground Telecommunications with the Software Connector, EUROCONTROL Experimental Centre (EEC), in the 22nd *Digital Avionics Systems Conference*, 12 to 16 October, pp. 6.A.3 to 61–12, IEEE.

Fewell, M.P. & Hazen, M. G. (2003) *Network-Centric Warfare – Its Nature and Modelling*, DSTO-RR-0262, Maritime Operations Division, Defence Science & Technology Organisation, Edinburgh, Australia.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley.

Heineman, G. & Councill, W. (2001) *Component-Based Software Engineering Putting the Pieces Together*, Addison-Wesley.

Henning, M. & Vinoski, S. (1999) *Advanced CORBA Programming with C++*, Addison-Wesley Professional.

Hill, R. (2003) Network Centric Warfare, address to *ADF Network Centric Warfare Conference*, Canberra, Australia, 20 May (accessed 19 March 2007), URL: http://www.minister.defence.gov.au/HillSpeechtpl.cfm?CurrentId=2770

Hue, M. (2007) *A Methodology and Metrics for Evaluating NCW Implementation*, DSTO-RR-0324, Command and Control Division, Defence Science & Technology Organisation, Edinburgh, Australia.

Huston, S., Johnson, J. & Syyid, U. (2003) *The ACE Programmer's Guide: Practical Design Patterns for Network Systems Programming*, Addison-Wesley Professional.

Keus, H. E. (2005) Netforce Principles: An Elementary Foundation of NEC and NCO, in the *10th International Command and Control Research and Technology (CCRT) Symposium*, McLean, Virginia, U. S., 13 to 16 June, U. S. Department of Defense Command and Control Research Program.

Knight, M., Vencel, L. & Moon, T. (2006) *A Network Centric Warfare (NCW) Compliance Process for Australian Defence*, DSTO-TR-1928, Defence Science & Technology Organisation, Edinburgh, Australia.

Krause, M. (2005) The Case for Minimum-Mass Tactics in the Australian Army, *Australian Army Journal*, **II**(2), Land Warfare Studies Centre, Canberra, Australia, pp. 69–79.

Krishnamurthy, L. (2006) *Comparative Assessment of Network-Centric Software Architectures*, Masters thesis, Virginia Polytechnic Institute and State University.

Lawrie, G., Capon, S., Cutler, P., Filippidis, A., Iob, M., Pearce, B., Priest, T., Rockliff, S., Skinner, M., Temple, P., Tweedale, J. & White, K. (2005) Wedgetail Evolution: Soaring to Greater Heights?, in *Proceedings of the 11th Australian International Aerospace Congress*, Melbourne, Australia, 13 to 15 March, Engineers Australia.

Lea, D. (1999) *Concurrent Programming in Java: Design Principles and Patterns*, Addison-Wesley.

Lewis, G. & Wrage, L. (2004) *Approaches to Constructive Interoperability*, CMU/SEI-2004-TR-020, Carnegie Mellon University Software Engineering Institute, United States.

Logan, B. (2003) Technical Reference Model for Network-Centric Operations, *CrossTalk: The Journal of Defense Software Engineering*, August, pp. 21–5.

Lough, R. (2006) C2 and Australia's Approach to NCW, in the *C4I Asia Conference*, Singapore, 20 February, Asian Aerospace.

McKenna, T., Moon, T., Davis, R. & Warne, L. (2006) Science and Technology for Australian Network Centric Warfare: Function, Form and Fit, *Australian Defence Force Journal*, **170**, pp. 62–76.

Microsoft Developer Network (2007) *.NET Framework 3.0* (accessed 27 April 2007), http://msdn2.microsoft.com/en-us/netframework/default.aspx.

Moon, T. (2006) Are Networked and Net-Centric the Same?, in the *Defence Experimentation Symposium*, Sydney, Australia, 28 to 30 March, Defence Science and Technology Organisation.

MSDN (2007)—*see* Microsoft Developer Network (2007).

NCIOF (2005)—*see* Net-Centric Operations Industry Forum (2005).

Net-Centric Operations Industry Forum (2005) *Industry Best Practices in Achieving Service Oriented Architectures (SOA)*.

Object Management Group (2004) *Common Object Request Broker Architecture: Core Specification Version 3*, Object Management Group.

O'Brien, L., Bass, L. & Merson, P. (2005) *Quality Attributes and Service-Oriented Architectures*, CMU/SEI-2005-TN-014, Carnegie Mellon University Software Engineering Institute, United States.

Office of the Secretary of Defense (2001) *Network Centric Warfare: Department of Defense Report to Congress*, Department of Defense, Washington, DC, United States.

OMG (2004) — *see* Object Management Group (2004).

OSD (2001) — *see* Office of the Secretary of Defense (2001).

Overview of ACE (2006) (accessed 27 April 2007), http://www.cs.wustl.edu/~schmidt/ACE-overview.html.

Overview of CORBA (2006) (accessed 27 April 2007), http://www.cs.wustl.edu/~schmidt/corba-overview.html.

Paunicka, J., Mendel, B. & Corman, D. (2001) The OCP: An Open Middleware Solution for Embedded Systems, in the *Proceedings of the American Control Conference*, 25 to 27 June, vol. 5, pp. 3445–50, IEEE.

PGAD (2002) — *see* Policy Guidance and Analysis Division (2002).

Policy Guidance and Analysis Division (2002) *Future Warfighting Concept*, ADDP–D.02, Defence Publishing Service, Department of Defence, Canberra, Australia.

Rising, L. (ed.) (2001) *Design Patterns in Communication Software*, Cambridge University Press.

Schmidt, D., Stal, M., Rohnert, H. & Buschmann, F. (2000) *Pattern Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, vol. 2, Wiley.

Schmidt, D. & Buschmann, F. (2003) Patterns, Frameworks, and Middleware: Their Synergistic Relationships, in the *Proceedings of the 25th Software Engineering Conference*, 3 to 10 May, pp. 694–704, IEEE.

SoSECE (2007) — *see* System-of-Systems Engineering Center of Excellence (2007).

System-of-Systems Engineering Center of Excellence (2007) *SoS Engineering*, http://www.sosece.org/index.cfm?fuseaction=4F68D291-802C-E84F-68643FD05F21FFB4, (accessed 25 April 2007).

Szyperski, C. (1998) *Component software: Beyond Object-Oriented Programming*, Addison-Wesley.

US Army (2004) *Future Combat Systems 18+1+1 Whitepaper*, United States Department of the Army.

US Department of Defense (2004) Net-Centric Operations and Warfare Reference Model (NCOW RM), in *Defense Acquisition Guidebook* (accessed 3 April 2007), URL: http://akss.dau.mil/dag/GuideBook/IG_c7.2.1.4.asp.

US DoD (2004) — *see* US Department of Defense (2004).

US JFC (2001) — *see* US Joint Forces Command (2001).

US Joint Forces Command (2001) *Global Information Grid (GIG)*, Capstone Requirements Document JROCM 134-01, Norfolk, VA, United States.

Vencel, L. (2006) *Towards Establishing a Set of Australian DoD Net Centric Standards*, DSTO-CR-2006-0200, Defence Systems Analysis Division, Defence Science & Technology Organisation, Canberra, Australia.

Appendix A: AEW&C Mission System Testbed Components

Two types of components exist within the Airborne Early Warning & Control Mission System Testbed (AEW&C MST): (1) architectural components within the component-based software engineering (CBSE) development methodology discussed in Section 4.1 and (2) Commercial-Off-The–Shelf (COTS) components. The interfaces of architectural components conform to the component model of the Software Architecture Framework (SAF) and have been developed specifically for the AEW&C MST. These components can be grouped according to their roles of mission computing, stimulation and monitoring. The native interfaces of COTS components do not conform to the SAF component model and are identified separately.

A.1. Mission Computing

Mission computing components represent adaptations of architectural components that exist in the AEW&C Mission Computing Subsystem (MCS).

A.1.1 Track Manager

The Track Manager is responsible for maintaining the tactical state of the AEW&C MST. It does so through a repository of tracks and a collection of capabilities that can be applied to manage tracks. This behaviour can be compared to patterns associated with component technologies employing containers[13], in which components of a single type can be added, removed, located and returned. The track objects managed by the Track Manager have an interface defined in the Interface Definition Language (IDL), which provides independence from the programming language implementation and supports integration of separate applications and heterogeneous systems [Henning & Vinoski 1999].

The Track Manager interface supports the dynamic nature of tracks by allowing updates to occur on its stored collection. These updates are handled through a Fused Track Updater, utilising an Active Object pattern[14]. The Fused Track Updater separates the track repository update from the general execution of the Track Manger, permitting internal processing and requests from other clients to occur concurrently. The Active Object pattern is supported by the SAF using a Runnable object, which provides facilities for the execution of a Fused Track Updater through inheritance. The SAF Runnable object is similar in nature to a Java Runnable, which provides concurrent execution via a separate thread.

[13] Home in CORBA 3 [OMG 2004], EJBHome in Enterprise Java Beans [DeMichiel & Keith 2006], and Container in .NET [MSDN 2007].
[14] An Active Object '…decouples method execution from method invocation to enhance concurrency' [Rising 2001, p. 347].

A requirement of the Track Manager is to handle multiple update requests simultaneously. Simply being multi-threaded does not guarantee efficiency in the execution or the handling of these requests, leading to the employment of the Leader/Followers pattern as a hand-off strategy [Schmidt *et al.* 2000]. To accommodate the Leader/Followers pattern a thread pool is required. Although a Singleton [Gamma *et al.* 1995] thread pool is available through the SAF, a controllable one is useful in this application, leading to the instantiation of a thread pool within the scope of Fused Track Updater. The Leader/Followers pattern permits the Fused Track Updater's threads to take turns processing requests to improve the liveliness of the Track Manager throughout the update process.

To manage the lifecycle of a track object, the Track Manager requires mechanisms for the removal and deactivation of tracks from the repository. Data associated with a track object is volatile and dynamic and therefore becomes useless to the Track Manager if not updated periodically. These time-expired tracks are removed through a track eviction mechanism, which itself employs an Active Object pattern for parallel execution.

The Track Manager must also provide an interface that allows clients to access tracks that are managed and contained by the Track Manager. This interface encapsulates iteration over the track container, employing the Façade pattern [Gamma *et al.* 1995]. This provides independence from the Track Manager's internal representation of a track and ensures access occurs within the context of the Track Manager's internal read/write locking mechanisms. These locking mechanisms ensure the containment remains consistent despite the volatile nature of the data.

A.1.2 Rosetta Adapter

The AEW&C MST makes use of the COTS tactical gateway Rosetta, described in Section A.4.2, to interface to Tactical Data Link (TDL) networks. Use of this software reduces the development needs of the AEW&C MST, which otherwise would have incorporated coding of data link message sets and the handling of data from varying hardware interfaces. Rosetta isolates the user from these details, enabling programmers to focus on Rosetta's Real-Time Query Language (RQL), a language similar in many respects to the Sequential Query Language (SQL) associated with regular databases. RQL enables the integration of different TDLs into the experimental environment of the AEW&C MST. This forms the basis of two Rosetta clients, which interact with the rest of the AEW&C MST as shown in Figure 5–4.

The first Rosetta client allows data to flow from the AEW&C MST to the track containment of a remote Rosetta Server. Currently, this supports the transmission of the AEW&C's ownship data, providing information equivalent to that observed normally in a tactical situation. Extension of this interface will enable the transmission of other information, such as surveillance data.

The second Rosetta client permits data flow in the opposite direction, from a remote Rosetta server to the AEW&C MST. Through RQL statements, this client represents a source of tactical information for the AEW&C MST, on which the AEW&C MST can periodically poll for updated track data. To external systems the client represents a tactical information sink and facilitates the flow of data from potentially many systems to the AEW&C MST.

A.1.3 Ownship Status

The Ownship component maintains the kinematic state of the aircraft represented by the AEW&C MST. Access to this data is provided through an interface defined in IDL and updates occur through a callback. The Ownship employs the Home pattern to gain access to the component generating this kinematic data, which for example could be registered with Air Vehicle (Section A.2.2) to receive regular kinematic data updates.

A.2. Stimulation Environment

The stimulation environment components represent information sources that exercise scenarios with the AEW&C MST.

A.2.1 Track Source Adapter

The Track Source Adapter is an abstract entity for input sources. Track input sources typically come in many forms, which necessitates the transformation to a common behavioural model for the stimulation of the Track Manager (Section A.1.1). This is achieved through the use of a Strategy pattern[15]. Conforming to this pattern, a Track Generation Strategy provides a common abstract interface for defining a track, with the implementation of the track definition unique to the particular track source.

Track data is provided to the Track Manager through a separate class, a Track Writer, using an Active Object pattern. The Track Writer Active Object decouples requests to the Track Manager from any internal processing required on the part of the Track Source Adapter's concrete implementation. This processing is likely to maintain the state of those tracks under the concrete implementation's control, and is able to execute concurrently with the Track Writer to remove any dependencies on the possibly remote calls to the Track Manager.

The Test Track Generator provides a concrete implementation of the abstract Track Source Adapter for the generation and maintenance of random tracks. This component defines its own strategy, a Test Track Generation Strategy, for the creation of tracks. Its concrete implementation produces a defined number of tracks with random data. This random data is generated to conform to the common representation outlined by the Track Manager in Section A.1.1.

The Test Track Generator is responsible for the maintenance as well as the generation of random tracks, requiring local storage to maintain internal state. This storage is similar to that within the Track Manager and periodic updates occur to each stored track in

[15] A Strategy pattern can be categorised as a family of encapsulated algorithms that are made interchangeable [Gamma *et al.* 1995].

accordance with its randomly generated initial conditions. Data stored within the Track Manager's container must remain consistent with these updates and thus results in further requests made to the Track Manager. As such, those arguments made for the use of a Track Writer Active Object are valid here, leading to the creation of a Track Updater Active Object.

A.2.2 Air Vehicle Model

The Air Vehicle component is a simple model of an aircraft, maintaining aircraft position with basic limits on speed, altitude, acceleration, turn-rate and climb-rate. A pilot command interface is defined to control each of these attributes and thus the basic movement of the modelled aircraft, which can also be configured manually prior to deployment.

The Air Vehicle component is strictly stimulus for the mission computing elements of the AEW&C MST, therefore requiring the Ownship Status to be notified of any necessary updates to the state of the AEW&C MST. Consistent with other component requests, Air Vehicle contains an Updater that is an Active Object for this purpose. As the Updater itself does not contain any state data, the Home pattern is employed to access the relevant kinematic data within Air Vehicle.

A.2.3 STAGE

Distributed Information Simulation (DIS) is one method by which simulation can stimulate AEW&C MST experimentation. A DIS interface is provided for the AEW&C MST by a COTS product, the Stimulation Toolkit and Generation Environment (STAGE). STAGE is described in more detail in Section A.4.1. Currently version 5 has been, and will be further integrated into the AEW&C MST. Version 4 will be reverted to for user modules and simulation models received from the AEW&C program that are incompatible with the latest version of STAGE. STAGE version 5 provides an enhanced set of DIS Protocol Data Units (PDUs) over its predecessor, with the inclusion of the Electromagnetic Emission PDU, and supports extension to include any other type of PDU.

The STAGE interface to the AEW&C MST acts as a source of information for simulation entities. This enables the AEW&C MST to observe and interact with scripted platforms and events, all under the control of a scenario running internally within STAGE. Through its DIS interface, STAGE also has the ability to receive information about entities scripted by other external simulations, thus forming the basis for the AEW&C MST's inclusion in distributed simulations. Further development will automate the activation of the STAGE DIS interface, include additional simulation models and incorporate track data fusion.

A.3. Monitoring

Monitoring components provide interfaces for observing the information received and stored by the other components of the AEW&C MST.

The Track Monitor is a simple component that periodically accesses and displays the details of all tracks in the Track Manager to an operator. The Track Monitor itself is responsible for activating two Active Objects, each tasked with presenting alternative outputs.

Track Streamer is one of the Active Objects activated by the Track Monitor. Unlike other Active Objects described previously it does not incorporate the Home pattern, and therefore does not have any reference to the Track Monitor. The Track Streamer defines an output stream within the SAF as its means for output, which is a common interface for writing data. This output stream conforms to a push model for writing, indicating that it knows the identity of the receiver before pushing the message. A callback is defined to encapsulate the writing of tracks to this output stream.

TDFAdapter is the second Active Object activated by the Track Monitor. Like the Track Streamer is does not employ the Home pattern and does not have any reference to the Track Monitor. The TDFAdapter employs an Adapter Pattern as described in Section A.4 to provide a service oriented interface to the COTS Tactical Display Framework (TDF) (Section A.4.3). This adaptation is encapsulated within a callback to ensure the necessary data conversions take place for accurate representation on the TDF Graphical User Interface (GUI).

A.4. Commercial Off-The-Shelf

A COTS component is distinguished from those developed specifically for the AEW&C MST by its lack of conformance to a component model. Fitting the CBSE general description of a component, COTS components provide both behaviour and coordination, thus specifying not only what a component does but also how it interacts. However, a COTS component implements these details in a way that is unique to the product and not bound by component interactions or other architectural constraints [Bachmann *et al.* 2000].

COTS components are generally, but not always, produced by a third party. This makes editing the native interfaces difficult. However, an adapter class can be written to translate the proprietary nature of the software. An adapter allows access to the desired behaviour of a component without necessarily using its expected interface and, in experimentation with the AEW&C MST, normalises the behaviour of the COTS component to adopt a Service Oriented Architecture (SOA) approach. This SOA approach leads to the production of a service that conforms to the layered communication model discussed in Section 4 and preserves the existing capabilities of the component. This alternative interface interacts directly with the middleware layer and enables platform abstraction to be preserved. Examples of these adapters are discussed in Section A.4.2 and Section A.4.3 as applied to Rosetta and TDF respectively.

A.4.1 STAGE

STAGE is a software tool that supports many utilities in the tactical domain. Its uses include training and evaluation, analysis of tactical scenarios and systems, and the simulation of real and synthetic systems. To support these utilities STAGE provides the facility to build and display real-time synthetic environments consisting of entities interacting through detection, engagement and destruction. STAGE is highly customisable with several techniques available for the extension of various aspects of the synthetic environment as well as its own simulation engine.

STAGE consists of a number of applications whose interactions with each other and internal and external data sources are shown in Figure A–1.

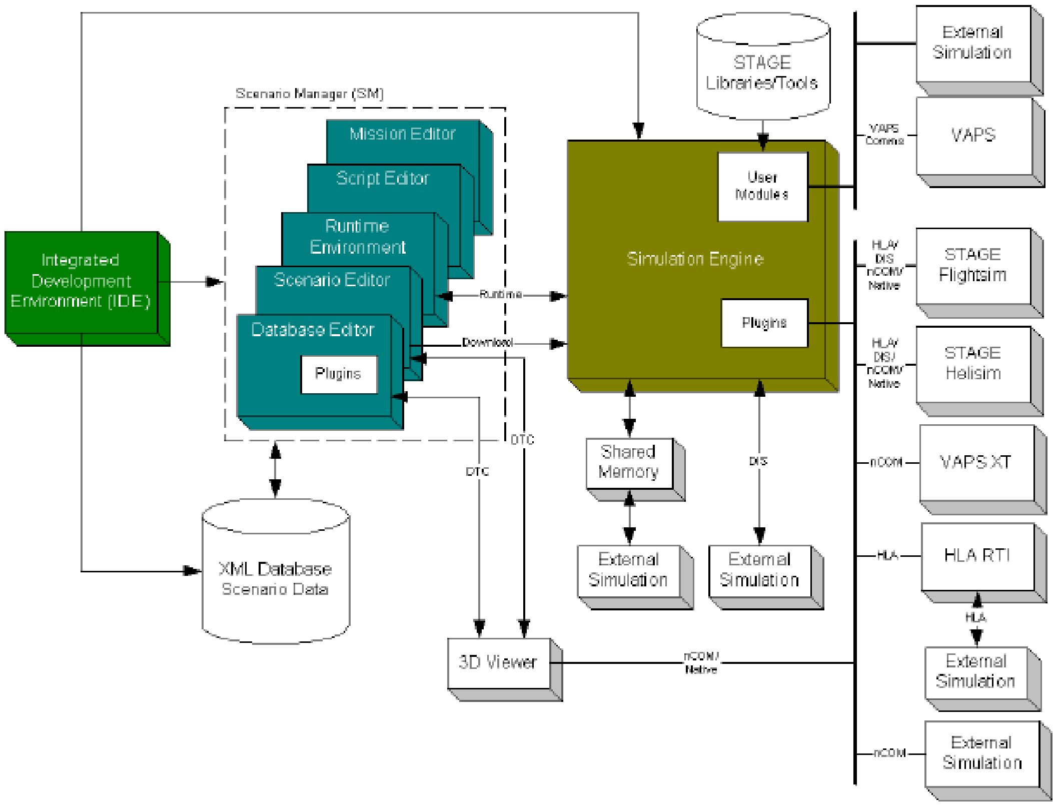

Figure A–1. Relationship of STAGE applications. Source: [CAE 2006]

The STAGE applications are described as follows:

- Scenario Manger (SM): Provides a number of environments to assist each phase of a scenario's lifecycle. The Scenario and Database editors handle the development and assembly of components for a scenario, while the Runtime Environment monitors and controls its execution. Embedded within the Runtime Environment is the Situation Awareness Display (SAD), which is responsible for visualising the simulation.

Additional behaviour and extensions to scenario elements are managed through the script and mission editors.

- Simulation Engine (SIM): The simulation of the synthetic environment. It includes a simulation engine and modules that model the behaviour of entities within the synthetic environment.

- Integrated Development Environment (IDE): An interface allowing for the modification of simulation data structures.

- Map Generator (Genmap): Converts maps from different data formats into STAGE's proprietary format.

- Logger: Supports scenario review through recording and playback facilities.

A.4.2 Rosetta

Rosetta is a software package that allows processing, translation and forwarding of real-time sensor, navigation and data link data. To accomplish and assist this objective a number of applications are provided, each discussed below.

The Rosetta engine can be subdivided into individual software modules. These modules include a text parsing engine, RQL and a Forwarding Rules Object Gateway (FROG).

The text parsing engine eliminates the need for hard coding by defining Interface Control Documents (ICDs) as plain ASCII files. These are parsed on initialisation to generate a normalised form inside a Real-Time Track Database; a store for all tactical data encountered by the system. Track data in the database is accessed using RQL. This permits users to request details on any track or command via queries on the common field names. The FROG handles forwarding from one tactical data link message set to another, conforming to rules generated from the relevant standards.

The Joint Moving Map Tactical Information Display System (JMMTIDS) is a command and control (C2) GUI displaying in real-time and providing control over information from a variety of data links. It also supports a mission playback facility, where data can be recorded and then played back with various timing controls.

The Rapid Loader enables users to control Link 16 terminals through network initialisation loads and a variety of diagnostic tools. The Rapid Loader can also act as a MIL-STD-1553[16] bus controller, providing further control of terminal characteristics such as JTIDS[17] Unit (JU) number.

The Scenario Generation Toolset (SGT) is a data link scenario generator capable of creating a scenario incorporating multiple links in an offline GUI. SGT also supports the capability to add link data into the Rosetta database in real-time.

[16] A military standard that defines the electrical and protocol characteristics of a data bus.
[17] Joint Tactical Information Distribution System.

A.4.3 Tactical Display Framework

The TDF is a Java based and machine independent tactical display system designed for applications such as C2 and air traffic management. It is highly adaptable at both a user and application level, with features easily enabled or disabled for particular operators, and custom plug-ins added to change or provide new functionality. From a tactical perspective, a variety of mapping products are supported, as are multiple symbol sets and hooking capabilities.

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION **DOCUMENT CONTROL DATA**	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE Exploring a Net Centric Architecture using the Net Warrior Airborne Early Warning and Control Node	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) Kate Foster, Adam Iannos, Geoff Lawrie, Peter Temple and Brad Tobin	5. CORPORATE AUTHOR DSTO Defence Science and Technology Organisation PO Box 1500 Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER DSTO-TR-2093	6b. AR NUMBER AR-014-085	6c. TYPE OF REPORT Technical Report	7. DOCUMENT DATE December 2007

8. FILE NUMBER 2007/1036634/1	9. TASK NUMBER 07/044	10. TASK SPONSOR AEW&CSPO	11. NO. OF PAGES 61	12. NO. OF REFERENCES 63

13. URL on the World Wide Web http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-2093.pdf	14. RELEASE AUTHORITY Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS

network centric warfare; airborne mission systems; experimentation; test beds; software architecture; middleware; object-oriented system architecture; interoperability

19. ABSTRACT

Network Centric Warfare experimentation is required in order to transform the Australian Defence Force into a net centric force. One area of experimentation is net centric software architectures, particularly component-based systems and middleware. The Airborne Early Warning & Control Mission System Testbed (AEW&C MST) enables such experimentation to be conducted and is overviewed in this report. The AEW&C MST is also one node in the Net Warrior Initiative, which aims to conduct net centric experimentation with real systems, testbeds and simulators across DSTO. This report discusses Net Warrior and the role of the AEW&C MST as the AEW&C node.